Universidade de Aveiro
2016

Departamento de Eletrónica,
Telecomunicações e Informática

Marco António Silva Henriques

Reconhecimento facial com base em compressão de imagem

Facial recognition based on image compression

Universidade de Aveiro
2016

Departamento de Eletrónica,
Telecomunicações e Informática

Marco António Silva Henriques

Reconhecimento facial com base em compressão de imagem

Facial recognition based on image compression

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor António José Ribeiro Neves, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Armando José Formoso de Pinho, Professor associado c/ agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais pelo seu esforço e apoio.

o júri / the jury

presidente / president

Prof. Doutor Manuel Bernardo Salvador Cunha
Professor Auxiliar, Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Luís Filipe Pinto de Almeida Teixeira
Professor Auxiliar, Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor António José Ribeiro Neves
Professor Auxiliar, Universidade de Aveiro

agradecimentos /
acknowledgements

Em primeiro lugar gostaria de agradecer ao meu orientador, Professor Doutor António José Ribeiro Neves, pela sua total disponibilidade, ajuda e confiança, e ao meu co-orientador, Professor Doutor Armando José Formoso de Pinho pela sua ajuda e supervisão.
Aos meus pais, pela oportunidade que me deram de hoje estar onde estou, por todo o apoio e esforço que fizeram por mim.
Aos colegas e novos amigos que fiz ao longo da minha vida universitária, com quem partilhei grandes momentos que levarei para a vida.
Aos amigos de longa duração que, apesar do passar dos anos, sempre estiveram/estarão presentes quando necessário.
À minha família, pelas contantes manifestações de apoio e confiança.
Aos docentes que contribuiram, não só para o meu desenvolvimento intelectual, mas também pessoal.
Todos vocês deram o seu contributo para o meu sucesso académico e para a vida. Por isso dirijo a todos a minha muito sincera palavra de apreço e agradecimento.

Palavras Chave

Reconhecimento facial, processamento de imagem, semelhança entre imagens, compressão de dados, informação, Modelos de Contexto Finito, estatística.

Resumo

O reconhecimento facial tem recebido uma importante atenção em termos de investigação, especialmente nos últimos anos, podendo ser considerado como uma das mais bem sucessidas aplicações de análise e "compreensão" de imagens. Prova disso são as várias conferências e novos artigos que são publicados sobre o tema. O foco na investigação deve-se à grande quantidade de aplicações a que pode estar associado, podendo servir de "auxílio" para muitas tarefas diárias do ser humano.

Apesar de existirem diversos algoritmos para efetuar reconhecimento facial, muitos deles até bastante precisos, este problema ainda não está completamente resolvido: existem vários obstáculos relacionados com as condições do ambiente da imagem que alteram a aquisição da mesma e que, por isso, afetam o reconhecimento.

Esta tese apresenta uma nova solução ao problema do reconhecimento facial que utiliza métricas de similaridade entre imagens, obtidas com recurso a compressão de dados, nomeadamente a partir de Modelos de Contexto Finito. Existem na literatura algumas abordagens ao reconhecimento facial através de compressão de dados que recorrem principalmente ao uso de transformadas. O método proposto nesta tese tenta uma abordagem inovadora, baseada na utilização de Modelos de Contexto Finito para estimar o número de bits necessários para codificar uma imagem de um sujeito, utilizando um modelo de treino de uma base de dados.

Esta tese tem como objectivo o estudo da abordagem descrita acima, isto é, resolver o problema de reconhecimento facial, para uma possível utilização num sistema de autenticação real. São apresentados resultados experimentais detalhados em bases de dados bem conhecidas, o que comprova a eficácia da abordagem proposta.

Keywords

Facial recognition, image processing, image similarity, data compression, information, Finite Context Models, statistics.

Abstract

Facial recognition has received an important attention in terms of research, especially in recent years, and can be considered as one of the best succeeded applications on image analysis and understanding. Proof of this are the several conferences and new articles that are published about the subject. The focus on this research is due to the large amount of applications that facial recognition can be related to, which can be used to help on many daily tasks of the human being.

Although there are many algorithms to perform facial recognition, many of them very precise, this problem is not completely solved: there are several obstacles associated with the conditions of the environment that change the image's acquisition, and therefore affect the recognition.

This thesis presents a new solution to the problem of face recognition, using metrics of similarity between images obtained based on data compression, namely by the use of Finite Context Models. There are on the literature some proposed approaches which relate facial recognition and data compression, mainly regarding the use of transform-based methods. The method proposed in this thesis tries an innovative approach based on the use of Finite Context Models to estimate the number of bits needed to encode an image of a subject, using a trained model from a database.

This thesis studies the approach described above to solve the problem of facial recognition for a possible use in a real authentication system. Detailed experimental results based on well known databases proves the effectiveness of the proposed approach.

Contents

List of Figures

List of Tables

Glossary

FCM	Finite Context Models
CCTV	Closed-Circuit TeleVision
PCA	Principal Component Analysis
LDA	Linear Discriminant Analysis
JPEG	Joint Photographic Experts Group
GIF	Graphics Interchange Format
HDTV	High-Definition Television
MSE	Mean Squared Error
PSNR	Peak Signal-to-Noise Ratio
DCT	Discrete Cosine Transform
LBP	Local Binary Patterns Histograms
JPEG-LS	Lossless JPEG
LOCO-I	LOw COmplexity LOssless COmpression for Images
PNG	Portable Network Graphic

CHAPTER 1

INTRODUCTION

Image compression is a way to reduce what is called the redundancy of image data, i.e., information that an image may have in excess, or it is not very relevant (for the human eye) in terms of the image itself. On many cases, this excess of information is related to the human's visual system: sometimes when compression is performed on an image, although some information was "thrown" away, it looks exactly the same for the human eye. Image compression can be very useful, since it is an efficient form to store or transmit data (Figure 1.1).

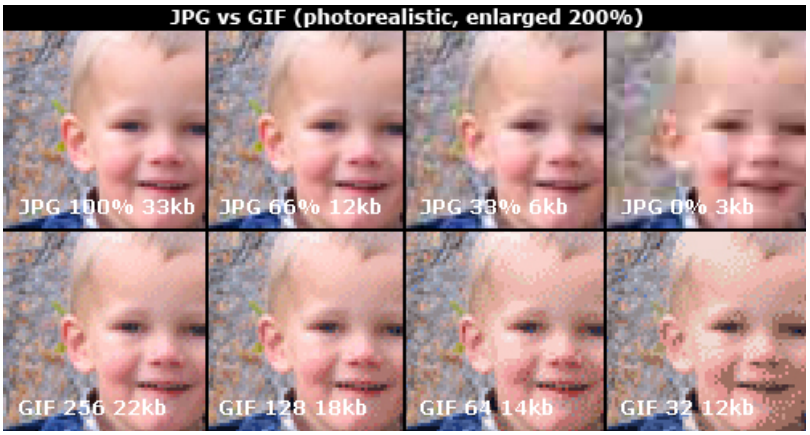


Figure 1.1: Image compression example; quality and size of the images are presented [48].

On other hand, there is a problem which has been studied since the beginning of computer vision. This problem is the facial recognition (Figure 1.2).

It is quite easy for a human to recognize another human being, independently on weather conditions, distance (limited of course)...but is it that easy for a computer? Well, the task to confirm that a person is who claims to be, or to recognize (identify with a label/name) a certain person, can be a really hard task for a computer, because there are several obstacles which can be related to it. This problem shall be explored on this thesis.

Can these two subjects described above be related? In other words, can compression based algorithms be used to perform facial recognition?

Figure 1.2: Facial Recognition example [43].

This is the question that is to be discussed in the present document.

This thesis presents a study on the use of compression to verify similarity between images; in general, when a compression method is performed, one gets a number or metric at the end regarding some compression aspect. In this case, the output of the used compression algorithm is the number of bits to encode a certain image, therefore it will be used as a metric of similarity between images.

In particular, this work focuses on the use of this similarity metric for the problem of face recognition.

1.1 MAIN CONTRIBUTIONS

The main goal of this work was to study, develop, and evaluate the use of compression algorithms to be used on facial recognition. As a result of this work, there are some contributions:

- Implementation of a Finite Context Models class.
- Implementation of a Predictive Methods class.
- Software for face recognition based on compression.
- Detailed experimental results of recognition rates for the case of three distinct databases.

1.2 THESIS STRUCTURE

This document is structurally divided into the following chapters:

- Chapter 2 is dedicated to the introduction of the facial recognition problem. Several methods are presented to perform it, based on how the facial data is collected. We present a study using some well known 2D algorithms on the subject.
- Chapter 3 presents a review on data compression, which is the base of the work, describes different ways to achieve compression, including the method used to perform facial recognition, Finite

Context Models. There is also a small section about predictive methods, another compression algorithm, which possible contribution to facial recognition was also studied. The practical implementation is also explained.

- Chapter 4 contains the results of this work, which consists on the use of compression to facial recognition purposes, namely using Finite Context Models, on different environment conditions. Basically, the recognition rate using the FCM library is presented. Three distinct databases were used.
- Chapter 5 includes general conclusions about the developed work, as well as possible ideas and suggestions on future complementary work on the same subject.
- Appendix A presents the detailed results, i.e., all the obtained plots from the results.
- Appendix B is a user's manual, explains how the FCM library (as well as other programs) can be used to perform recognition.

CHAPTER 2

Face Recognition

"As one of the most successful applications of image analysis and understanding, face recognition has recently received significant attention, especially during the past several years. At least two reasons account for this trend: the first is the wide range of commercial and law enforcement applications, and the second is the availability of feasible technologies after 30 years of research. Even though current machine recognition systems have reached a certain level of maturity, their success is limited by the conditions imposed by many real applications.For example, recognition of face images acquired in an outdoor environment with changes in illumination and/or pose remains a largely unsolved problem. In other words, current systems are still far away from the capability of the human perception system." [4]

This chapter is dedicated to the introduction of the facial recognition problem, its objectives, possible uses and difficulties. We present some types of algorithms used, where three of them will be reviewed: EigenFaces, FisherFaces and Local Binary Patterns Histograms (LBP).

2.1 why face recognition?

When we daily look at people, soon we realize that our eyes first fall into the face of these same people in order to seek their identity: "This is individual X". The facial appearance is a strong attribute, allowing us to easily recognize a given person.

Face recognition is used for two primary tasks:

1. Verification: When presented with a face image of an unknown individual along with a claim of identity, ascertaining whether the individual is who he/she claims to be (one-to-one matching) (Figure 2.1);
2. Identification: Given an image of an unknown individual, determining that person's identity by comparing that image with a database of images from known individuals (one-to-many matching) (Figure 2.2).

There are several real life applications where these two tasks of facial recognition may be used [1]:

- Security (access control to buildings, airports/seaports, ATM machines and border checkpoints; computer/network security);

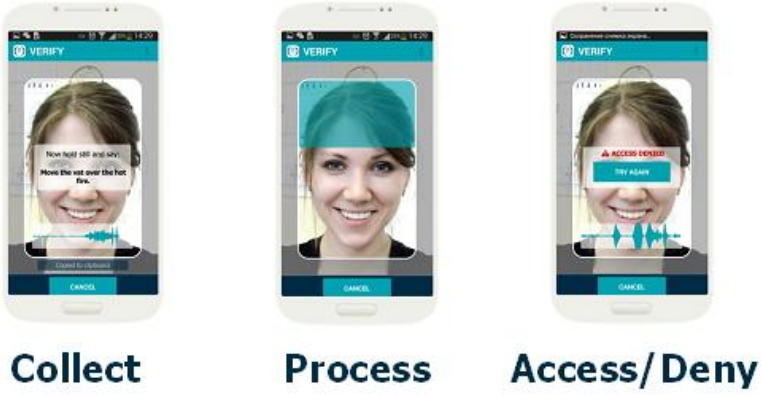


Figure 2.1: Facial verification example [49].

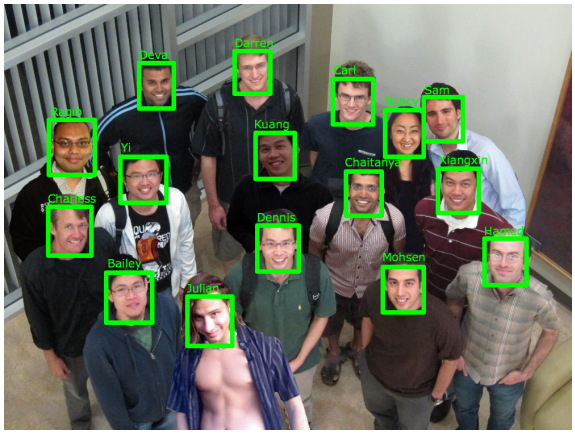


Figure 2.2: Facial identification example [50].

- Surveillance (a large number of CCTV can be monitored to look for known criminals, drug offenders, etc. and authorities can be notified when one is located);
- Criminal justice systems (mug-shot/booking systems, forensics);
- Image database investigations (searching image databases of licensed drivers, missing children, immigrants, police bookings, etc);
- General identity verification (electoral registration, banking, etc).

Thus, the importance of facial recognition is obvious, and one can conclude why the scientific community has been focused on the development of computational models to perform it. These computational models have been developed over the past years, making great progress on recognition success rates, as the main goal is to approximate the computer's facial recognition system to the human visual system. Unfortunately, facial recognition performed by a computer still has some problems, which do not affect the recognition in the case of a human:

- The faces are complex, multidimensional and its expression may not be neutral (facial expressions are the humans method to express what they feel);
- Large light variations;
- The capture angle of the face.

These are some of the largest obstacles of a facial recognition algorithm, because of the obviously alteration on the images, which can lead to wrong or non-recognition of a person.

Face recognition is a vividly researched area in computer science. First attempts were made in early 1970-ies, but a real boom happened around 1988, parallel with a large increase in computational power. The first widely accepted algorithm of that time was the Principal Component Analysis (PCA).

2.2 FACE RECOGNITION TECHNIQUES

Face recognition techniques can be divided into three categories, based on the face acquisition methodology: techniques that operate on still (intensity) images; those that deal with video sequences; and those that require other sensory data such as 3D information or infra-red images.

2.2.1 METHODS BASED ON INTENSITY IMAGES

Facial recognition methods based on intensity images fall into two main categories [1]:

- Holistic matching methods;
- Feature-based (structural) matching methods.

HOLISTIC MATCHING METHODS

Holistic matching methods make use of the whole face to perform recognition; generally involve the use of transforms to make the recognition robust to slight variations in the image. It is known that there are significant statistical redundancies in natural images [2]. This redundancies are even greater on a set of objects (facial images on this case) with no differences in terms of image transformations like scale, rotation or translation.

One example of an holistic algorithm is the EigenFaces method (based on PCA representation), which even today is used, not only as a benchmark method to compare new methods to, but as a base for many methods derived from the original idea. Another method is the FisherFaces (based on Linear Discriminant Analysis (LDA)). Both will be presented with some detail later in this chapter. One advantage of using a vector representation (such as PCA) is its reduced sensitivity to noise (like blurring, changes in background) as long as the topological structure remains the same, as shown in [4].

FEATURE-BASED (STRUCTURAL) MATCHING METHODS

Feature-based methods are methods based on the features of the images, instead of the whole image. They rely on the identification and extraction, for further processing, of the facial information, such as eyes, mouth, nose (along with its characteristics like position, size, like the example on Figure 2.3), thus computing geometric relations between them, which are then used by standard pattern recognition techniques to match (recognize) faces. There is an immediate advantage of these methods, which is the reduction of the input data to the algorithm.

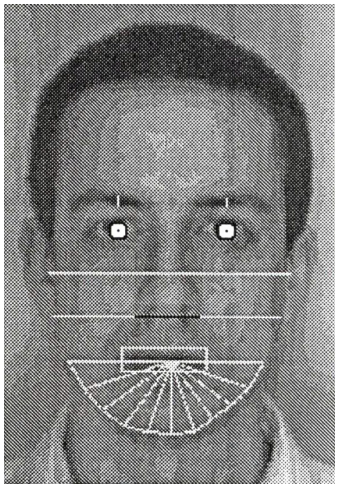


Figure 2.3: Facial recognition based on facial features [5].

A feature is an area of an image whose information is relevant to solve certain problems of computer vision (Figure 2.4). Examples of features are the corners, objects or edges. It is much easier, from the developer point-of-view, to extract information from an image by its features than trying to get it directly from its pixels, in addition to being much faster computationally.

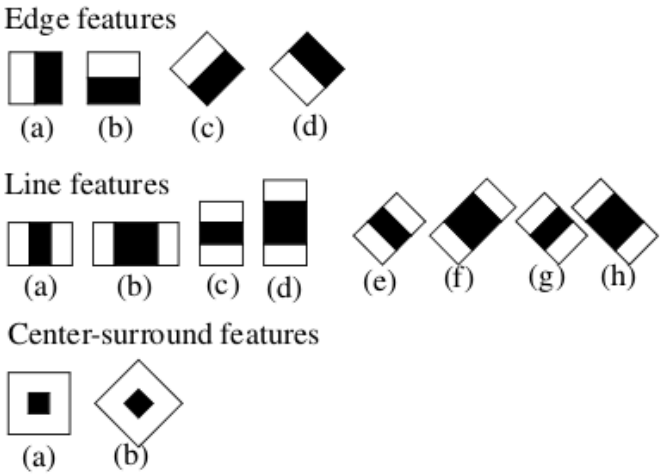


Figure 2.4: *Haar*-like features used on face recognition [6].

Due to the uniformity of shadows of the human face, it is easy to understand why these features (Figure 2.4) are used in facial detection and recognition, as can be seen on Figure 2.5.

Early work on facial recognition was based mainly on feature extraction methods. These methods are used by many facial recognition systems in addition to an holistic method (for example, the EigenFaces need accurate location of the facial features) [22], [23].

Figure 2.5: Usage of *Haar features* on a face [44].

ADVANTAGES AND DISADVANTAGES

One of the main characteristics of the holistic methods is the fact that they use the image as a whole, not concentrating on regions, which means they use all the image's information. This is, however, a "double-edge sword", because if it is an advantage the fact they do not discard any information, at the same time it is assumed that all the pixels of the image are relevant; this leads to computationally expensive techniques [1].

An advantage already mentioned is their good performance even with some image noise. On the other hand, they do not perform well on images with global variations like scale, illumination, etc; however, some of them have been enhanced to perform well on these cases. As a result, these approaches appear to produce better recognition results than the feature-based ones, in general.

One of the advantages of feature-based techniques is that, since the extraction of the feature points precedes the analysis done for matching, such methods are relatively robust to position variations in the input image. In principle, feature-based schemes can be made invariant to size, orientation and/or lighting. Another benefit is the high speed matching.

The major disadvantage of these approaches is the difficulty of automatic feature detection and the fact that the implementer of any of these techniques has to make arbitrary decisions about which features are important. If the feature set has no (or has few) relevant information in terms of discrimination ability, no further processing will compensate that fact [4].

Because there is not really an advantage of a type of method over the other, there are still some hybrid methods to the face recognition problem, using both holistic and local features approaches [22]–[24].

2.2.2 METHODS BASED ON VIDEO

On some facial recognition applications, the images of the subject to be recognized are not still. Instead, they are taken from image sequences (video example on Figure 2.6), which is the example of surveillance. So, in addition to a simple detection and recognition process (for example, on a picture), a video-based face recognition system also tracks the face over the time [8], [9].

A video based method can provide several advantages over still image methods [4], [25]:

- Good frames can be selected on which to perform the recognition;
- Video provides temporal continuity; this allows the reuse of recognition information obtained from good quality images to process low quality frames, resulting in improved recognition;

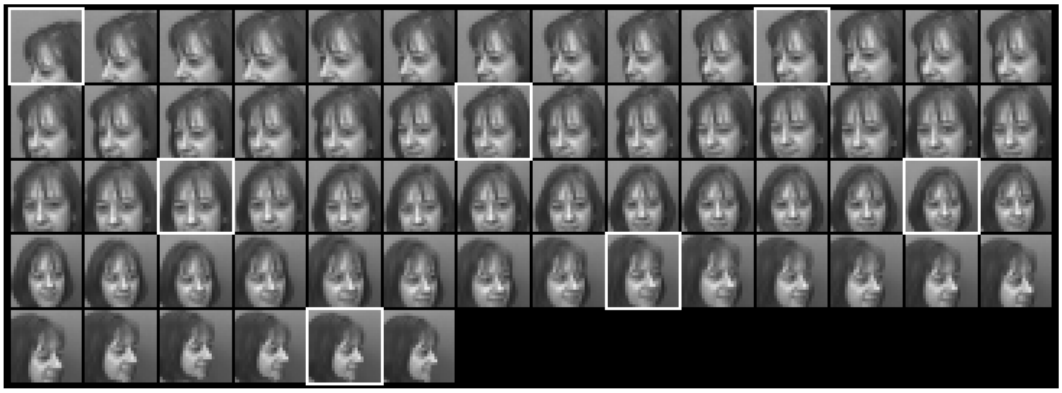


Figure 2.6: Facial recognition based on video [10].

- Video allows tracking of images; this way facial expressions or poses (for example) can be compensated.

However, facial recognition on video still has some problems to be dealt with:

- Low video quality: there are several unwanted situations that may occur during the video acquisition, because, in most cases, the environment is not controlled. Examples of this situations are changes on lighting, pose and scale or occlusion of faces;
- Still related to the video acquisition, the images taken are usually smaller compared to intensity image-based methods. Smaller images are obviously harder to recognize for a computer vision algorithm.

Some of the recognition methods from a video sequence can be seen as an extension to image based-recognition (from intensity images), but there are already some algorithms that use both spatial and temporal information.

2.2.3 METHODS BASED ON OTHER SENSORY DATA

Due to technological advances, from a few years ago, there has been some attention on using different types of image acquisition, namely 3D (scanning systems, structured light systems (Figure 2.7), stereo vision systems) or infra-red images. With the improvement of technologies, 3D face recognition has been increasing on this decade, where there have been/are appearing several recent articles on this area [11]–[13].

The main advantage of the 3D-based approaches is that the 3D model retains all the information about the face geometry. Moreover, 3D face recognition also can be proved to be a further evolution of the 2D recognition problem, because a more accurate representation of the facial features leads to a potentially higher discriminating power [15].

This techniques may be more useful on a one-to-one matching (verification), because typically a 3D-based approach is more expensive in terms of memory and computational times, so it may not be practical to implement on a one-to-many matching (for example, research on big databases).

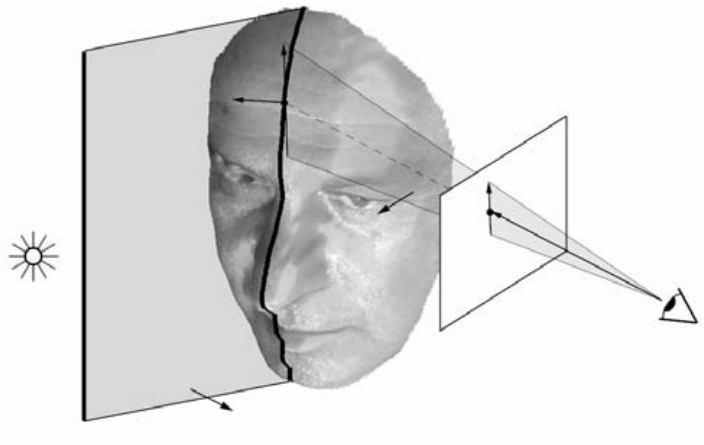


Figure 2.7: 3D facial recognition: structured light acquisition scheme [14].

2.3 2D FACE RECOGNITION TECHNIQUES: EIGENFACES, FISHERFACES AND LBP

We performed a study regarding some algorithms, namely the EigenFaces, FisherFaces and LBP algorithms. These three algorithms are used on 2D intensity image facial recognition; the first two are holistic methods, i.e., they focus on the whole image, whereas the third one is feature-based.
Regarding EigenFaces and FisherFaces, they are not very recent, but they were used during some time on facial recognition. They are computationally fast, reasonably simple and accurate at constrained environments. The EigenFaces (based of the PCA) is still used as comparison with recent algorithms, as already mentioned. This work was based on the OpenCV library, because it has some useful tools to its realization.

2.3.1 EIGENFACES

The main objective of the authors of this algorithm was to create a fast, simple and accurate facial recognition system. The EigenFaces method [16] is based on an information theory (holistic approach), where there is a group of facial images which most relevant information is to be coded.

Considering a 256×256 facial image (8-bit gray-scale), it can be viewed as a vector of dimension 256^2 ("stacking" all the columns into one); equivalently, it can be seen as a point on a 256^2-dimensional space. A group of face images can be mapped in this huge space. These face images, being similar in overall configuration, are not randomly distributed in the referred space, thus can be described by a lower dimensional subspace. This is done by performing a Principal Component Analysis (PCA), which objective is to find the vectors that best account for the distribution of face images within the entire image space. These vectors define the subspace of face images, also known as face space. Recognition consists on the projection of a new image on this subspace.

Each of these vectors describes an image, and is a linear combination of the original face images. Because these vectors are eigenvectors of the covariance matrix of the face images and they are face-like in appearance (the vectors), they are referred to as eigenfaces [16].

Recognition is performed by projecting a new face into the face space.

ALGORITHM

Assuming a training set composed by M faces (c columns by l rows matrix, let us consider $N = l = c$ for simplicity), $X = x_1, x_2..., x_M$, the average face can be defined by Equation 2.1

$$\mu = \frac{1}{M}\sum_{i=1}^{M} x_i \ . \tag{2.1}$$

Each face differs from the average by the vector $(x_i - \mu)$.

The set of this difference vectors is then subject of a PCA, seeking the M orthogonal vectors, which best describes the distribution of the data. K of these M vectors are chosen ($K \leq M$), corresponding to the higher eigenvalues λ_n. These vectors v_k and scalars λ_k, $k = 1, 2, ..., K$ are eigenvectors and eigenvalues, respectively, of the covariance matrix given by Equation 2.2

$$C = \frac{1}{M}\sum_{i=1}^{M}(x_i - \mu)(x_i - \mu)^T = AA^T \ , \tag{2.2}$$

where matrix $A = [(x_1 - \mu)(x_2 - \mu)...(x_M - \mu)]$.

Matrix C has a size of $N^2 \times N^2$ so, calculating the N^2 eigenvectors may be a difficult task even for a good computer. However if the number of points in the image space is less than its dimension ($M \leq N^2$), there will be only $(M - 1)$ meaningful eigenvectors. This means that, solving the N^2-dimensional eigenvectors corresponds to first solving for the eigenvectors of an $M \times M$ matrix and then taking appropriate linear combinations of the face images [16].

Now, a new face is transformed into its eigenface components, i.e., projected into the face space, by calculating the weights obtained by Equation 2.3

$$\omega_k = v_k^T(X - \mu), \ \ k = 1, ..., K \ . \tag{2.3}$$

These weights form a vector $\Omega^T = [\omega_1\omega_2...\omega_K]$ which describes the contribution of each eigenface on the input image and represents which of the S face classes (number of subjects/persons on a database) best fits the test face. The simplest method is to find the face class k that minimizes the Euclidean distance $\epsilon_k = ||(\Omega - \Omega_k)||$, where Ω_k is a weight vector representing the kth face class. The face is recognized (associated to a certain class) when this lower distance ϵ_k is below a chosen threshold θ.

The projection of the new image can lead to four possibilities [16]:

- Near face space and near a face class;
- Near face space but not near a known face class;
- Distant from face space and near a face class;
- Distant from face space and not a known face class.

In the first case, an individual is recognized and identified; in the second case, the individual is unknown; in the last two cases, the image does not correspond to a face.

Note that the value of K, which represents the principal components, i.e., the number of eigenfaces, can be chosen to be lesser than the total number of face classes S and, still, it can produce an accurate representation of the entire training set [16].

2.3.2 FISHERFACES

The PCA which is the core of Eigenfaces method, although it is a strong way to represent the information (finding linear combinations of features that maximize the total variance of the data), does not follow a "class-based approach" i.e., it does not consider any classes and a lot of discriminative information can be lost, therefore, ratings can be impossible. In order to work around this problem, Sir R. A.Fisher created the LDA, which reduces dimensional information based on a class-specific approach. The idea is to keep the similar classes close and different classes as far as possible from each other in dimensional representation. So, this idea was recognized as possible to use on face recognition [17].

ALGORITHM

Let X be a random sample vector with samples drawn from M classes, with N images each,

$$X = X_1 X_2 ..., X_M$$

$$X_m = x_1 x_2 ..., x_N \ ,$$

where $m = 1, 2, ..., M$.

The dispersion matrix S_B (between-class) and S_W (within-class) on an image space are defined as (equations 2.4 2.5 respectively):

$$S_B = \sum_{m=1}^{c} N_m (x_m - \mu)(x_m - \mu)^T \tag{2.4}$$

$$S_W = \sum_{m=1}^{c} \sum_{x_i \in X_m} (x_i - \mu_m)(x_i - \mu_m)^T \ , \tag{2.5}$$

where μ_m is the mean image (obtained similarly to the EigenFaces case) of class X_m and N_m the number of samples (faces) of class X_m (consider $N_m = N$).

Fisher's algorithm attempts to find a projection $W_{opt} = [w_1 w_2 ... w_K]$ that maximizes the class separation criterion given by Equation 2.6

$$W_{opt} = argmax_W \frac{|W^T S_B W|}{|W^T S_W W|} \ , \tag{2.6}$$

where $w_j, \ \ j = 1, 2, ..., K$ is the set of generalized eigenvectors of S_B and S_W corresponding to the K largest eigenvalues $\lambda_j, \ \ j = 1, 2, ..., K$, meaning,

$$S_B w_j = \lambda_j S_W w_j \ . \tag{2.7}$$

(Note: the maximum number of K is $M - 1$ where M is the number of classes).

Found the best projection W, with corresponding weight vectors w_j, the decision is made the same way as by the EigenFaces algorithm, i.e., the Euclidean distances between the new projection (face) and the classes on the database are calculated.

The Fisherfaces method has a better performance than Eigenfaces, especially when the faces have large amounts of lighting variation, it is faster and uses less memory. Furthermore, it solves one of the Eigenfaces problem, minimizing the dispersion within the same class, in addition to maximizing it between different classes [17].

2.3.3 FACIAL DETECTION

Most of the times, the concept of facial recognition comes up attached to another concept, which is facial detection (Figure 2.8). For example, when there are facial images on a picture, one can actually know that a certain "object" is, in fact, a face (via a set of characteristics that define the face class). This is facial detection, which can be defined as the act of revealing the existence of faces on a certain input data, such as images or video.

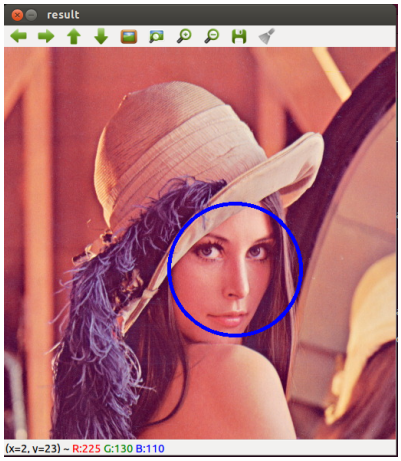


Figure 2.8: Example of facial detection on an image. The area corresponding to a face is evidenced but no label is associated to it (recognition).

This concept of detection is a really important one, because it usually precedes the recognition phase, meaning that if the detection is already inaccurate, the recognition algorithm will simply not work.

To perform facial detection (Section 2.3), we used the *Viola and Jones* algorithm [18]. Because of the relevance of the detection phase, this algorithm will be briefly described.

VIOLA AND JONES ALGORITHM

This detection algorithm is based on a machine-learning approach, where a function is trained by a large number of images, and used subsequently to detect objects (in this case faces) on other images [18], [19]. There are four concepts that are used by *Viola and Jones* algorithm and are crucial to its operation; they are:

- Image's features.
- Integral image.
- *AdaBoost*.
- *Cascade* of classifiers.

Image's features. This concept was already used and defined in this chapter, namely in Section 2.2.1.

The features used on *Viola and Jones* are the designated *Haar Features* that includes three types (Figure 2.9):

- Two-rectangle feature - the difference between the sum of the pixels in two similar rectangular zones, vertically or horizontally adjacent;
- Three-rectangle feature - the difference between the sum of the two outer rectangles and the central rectangle;
- Four-rectangle feature - the difference between diagonal pairs of rectangles.

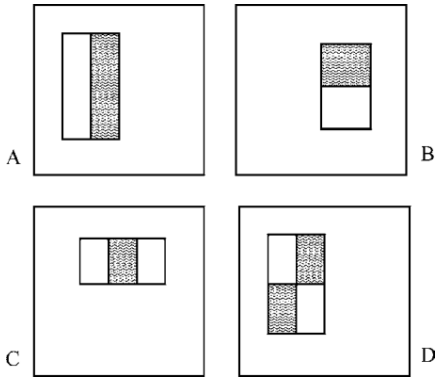


Figure 2.9: Examples of image features used in facial detection and recognition (*Haar features*) [18].

Integral Image. The integral image $I_{\sum}$ is an image where each pixel represents the sum of all the original image's pixels which are left and above it. Mathematically, pixel (x, y) value on the integral image is given by Equation 2.8

$$I_{\sum}(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \ . \tag{2.8}$$

Therefore, it is easy to see the relation between the calculation of the integral image and the features presented above: just get the first to directly calculate the second because now, an area of a certain feature is merely a difference between some pixels of the integral image. Thus, the number of operations can be reduced, because it is not needed anymore to calculate the sum of all pixels for each feature, which makes the computing time significantly lower. An example of the calculation of the integral image can be seen in Figure 2.10.

1	1	1
1	1	1
1	1	1

1	2	3
2	4	6
3	6	9

Figure 2.10: Calculation of the integral image (example). Left - original image; Right - Integral image.

Adaboost. Before talking about *AdaBoost*, it is important to define what is a classifier. A classifier can be seen as a system whose input is a set of measures/objects (faces in this case) and the output is a type of classification (either a name or a "yes/no" to their inclusion in a particular group). A classifier

is trained by providing a set of objects, called training set, which will cause the system to learn about it, such that, when given a new similar object, the classifier identifies as part of that same set.

In the present study, a classifier is trained with a set of face images, obtaining a mathematical model that allows determining a new image as a face or not.

AdaBoost is then a major boosting algorithm, or a combination of T simple classifiers (weak learners) to enhance (boost) their performance. This combination is given by Equation 2.9

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \ , \tag{2.9}$$

where α_t represents the relevance (weight) of each classifier $h_t(x)$. *AdaBoost* works as follows: First, apply all the features in the training images. For each feature, find the best threshold that classifies objects as positive (belongs to a certain class) or negative (does not belong to a certain class); select up the features that have lower error rate, i.e., that best classify the object. Here, it is an iterative process where, in each cycle, new errors and weights are calculated, repeated until a particular success rate is reached or all the features have been found.

A more detailed description of the algorithm can be checked below:

- Add the training set of N positive and negative objects, x_i, $i = 1, ..., N$ together with a label $f(x_i, y_i)$, where $y_i = 0$ for negative object and $y_i = 1$ for a positive one.
- Initialize the weights for different objects $w_{1,i} = \frac{1}{2m}$ or $w_{1,i} = \frac{1}{2l}$ for $y_i = 0$ or 1, where m and l are the number of positives and negatives, respectively.
- For $t = 1, ..., T$:
  1. Normalize the weights
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{N} w_{t,j}} \ .$$
  2. Select the single best classifier $h_t(x)$, in order to minimize the summed error ϵ_t to the boosted classifier obtained at the time t,
  $$\epsilon_t = \sum_i w_i |h(x_i) - y_i| \ . \tag{2.10}$$
  3. Update the object weights
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \tag{2.11}$$
  where $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$, $e_i = 0$ if x_i was correctly classified, $e_i = 1$, if wrongly classified.
- The final (strong) classifier is given by Equation 2.12
$$H(x)_= \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=0}^{T} \alpha_t \\ 0 & \text{others} \end{cases} \ , \tag{2.12}$$
where $\alpha_t = \log \frac{1}{\beta_t}$.

A major advantage of *AdaBoost* as a feature extractor relative to other algorithms of the same type lies within the learning speed [18].

Cascade of classifiers. The main problem of the previous described algorithm is the number of operations that have to be made to calculate all the features on an image; on *Viola and Jones* algorithm, a sub-window of 24x24 was used, which gives, approximately, a total of 160000 *Haar features* (as described above) [18]. To reduce the number of operations, the cascaded classifiers (Figure 2.11) were created, consisting of a sequence of boosted classifiers with increasing complexity, where the weak classifiers are first used. Thus, if a given image is rejected by a classifier, it is immediately dismissed and considered as not being part of the training set. This method leads to rapid rejection of many sub-windows.

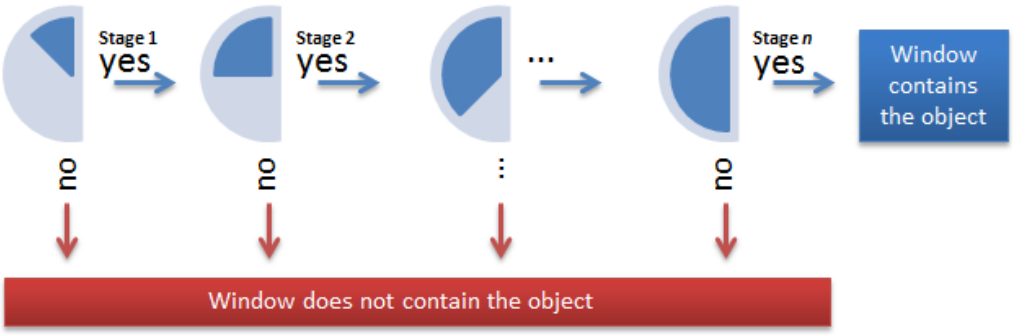


Figure 2.11: Cascade of classifiers. For a given object to be detected, it is necessary to pass all the stages of the cascade classifier [44].

2.3.4 EIGENFACES VS FISHERFACES

From a theoretical point of view, the FisherFaces method gives better results than EigenFaces when there are large brightness and facial expression variations [17]. The FisherFaces method is shown also to be faster and computationally lighter (less memory), being more accurate than the EigenFaces method. The EigenFaces method presents a solution against the failure of recognition due to these variations: remove the three main components of the image; although, this can prove to be a drawback, since this loss may include important information that helps facial recognition. Either way, the EigenFaces method does not only has disadvantages: it presents a better performance than FisherFaces if the database contains about seven or more different images for the same person (see Figure 2.12, results obtained resorting to the ORL database, which will be presented on the next Chapter).

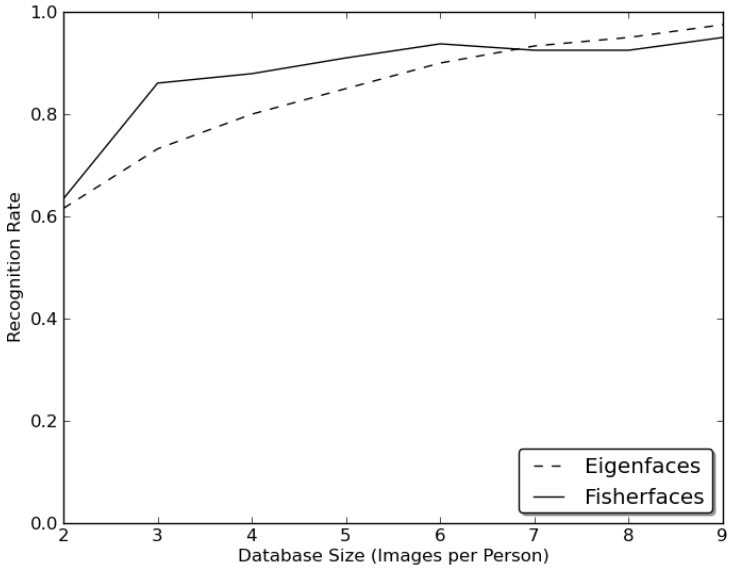


Figure 2.12: Success rate in facial recognition: EigenFaces vs FisherFaces, using the ORL database [45].

The results obtained in the study were performed resourcing to images taken from a video file (Figure 2.13). Although the input data is a video, there is no temporal information used: the images are treated frame-by-frame as if no relation existed between them.

Apart from that, in the video, light/scale conditions may vary randomly. Due to these aspects, the expected results are a bit lower (compared with the still-image ORL case). All images detected were resized to 100x100 (the algorithms require images with the same size).

Figure 2.13: Screenshots of the movie used to perform detection/recognition; three different situations may occur: correct detection and recognition (above), wrong detection (down left) and wrong recognition (down right).

The images were detected using *Viola and Jones* and recognized using EigenFaces and FisherFaces, using a previously stored database as the training set; the results were performed varying the size of this database, i.e., varying the number of training images per subject.

The results can be consulted in the table of Figure 2.14 and in the plot of Figure 2.15.

	Algorithm					
	FisherFaces			*EigenFaces*		
Faces	**Recognition (%)**	**Average Time (ms)**	**Total Time (s)**	**Recognition (%)**	**Average Time (ms)**	**Total Time (s)**
3	63.4	0.59	0.79	58.1	4	5.3
4	67.2	0.57	0.74	64.1	8.6	6.5
7	67.1	0.66	0.83	71.0	8.1	9.3
10	80.1	0.7	0.91	79.8	8	10.6

Figure 2.14: Recognition rates using different number of training images per subject.

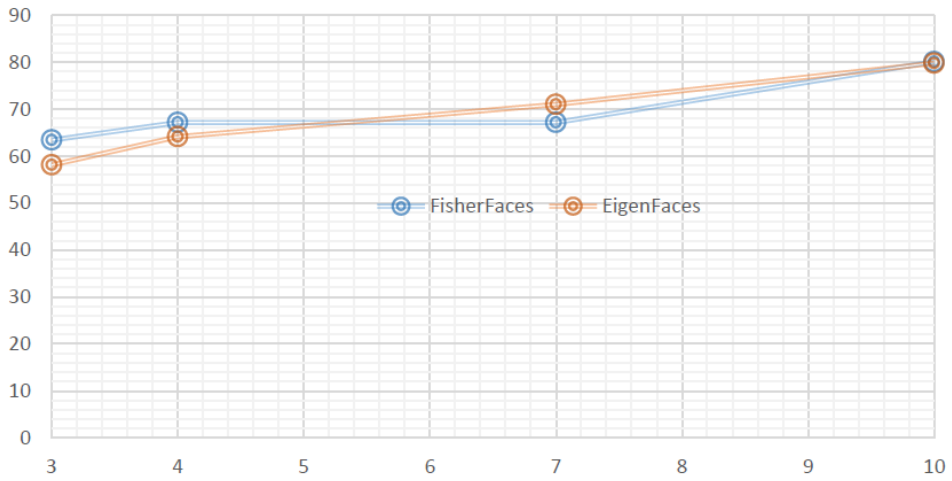


Figure 2.15: Experimental success rate in facial recognition: Eigenfaces vs Fisherfaces.

As one can see by the practical results on the plot, initially the EigenFaces had a lower recognition rate, which became less pronounced as the size of the database increased. At the end, the two algorithms present very similar results in terms of recognition, but the FisherFaces had the advantage of a significantly lower computing time.

2.3.5 LOCAL BINARY PATTERNS HISTOGRAMS (LBP)

Another algorithm included in the OpenCV library, which can be used to perform facial recognition, is the LBP algorithm [20]. This algorithm is a feature-based one: it summarizes the local structure of an image comparing each pixel with its neighborhood. All the pixels have a binary value, which is obtained comparing its intensity with each of the eight neighbor pixel's ones. In the example of Figure 2.16, starting in the pixel placed northwest from the central one and going clockwise (most to less significant bit), the binary value is given as follows:

- Bit takes value 1, if the central pixel's intensity is greater or equal than its neighbor;
- Bit takes value 0, in the other case.

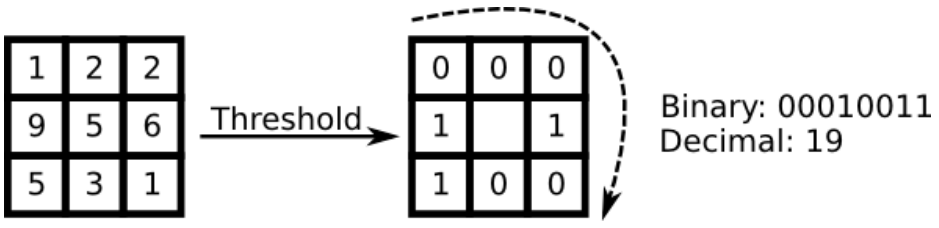


Figure 2.16: LBP algorithm example. Original LBP operator [46].

To use neighborhoods of different sizes, the LBP operator was extended by drawing circles with radius R from the central pixel [21].

So, consider P is the number of pixels in the neighborhood and R the size of the radius. The value of the pixel using the LBP algorithm, i.e., the LBP operator, is given by Equation 2.13

$$LBP_{P,R}(x,y) = \sum_{p=0}^{P-1} s(g_p - g_c)2^P \ , \tag{2.13}$$

where the function $s(x)$ corresponds to the bit values mentioned above, g_c is the central pixel intensity and g_p represents the p-neighbor intensity.

This method allows to unveil some details on images. See Figure 2.17 for some examples of possible captured neighborhoods.

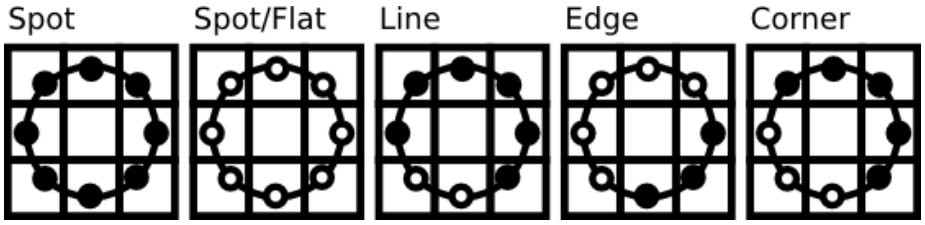


Figure 2.17: Possible neighborhoods for a central pixel [46].

The images can be divided onto several blocks. The values of the LBP operator can be put into an histogram (one per block), basically looking at statistics, i.e., how many times a certain number appeared. Because the number of intensities is only 256, these statistics are actually quite robust. These histograms can then be directly compared as some form of similarity between images, so, possible to apply to facial recognition.

Figure 2.18 shows the detection and recognition rates, as well as computational times of the three algorithms mentioned, using the referred video for testing.

Algorithm	Detected Subjects	Correct Detections	Detection (%)	Recognition (%)	Total prediction time (s)	Average prediction time (ms)	Total number of processed faces
LBP	36	28	77.8	78.6	79,6	59	1350
FisherFaces	37	28	75.7	80.1	0.9	0.7	1285
EigenFaces	39	29	74.4	79.8	10.6	8	1320

Figure 2.18: Detection/Recognition rates and processing times achieved with the three algorithms.

There are some extensions to LBP which allow better results in terms of translations and rotations of the test images [26], [27].

2.4 FINAL REMARKS

In this chapter, the facial recognition problem was introduced, as well as relevant information on the subject, like its possible applications and different ways to perform it.

In terms of practical results, the recognition rate achieved was approximately the same on the EigenFaces, FisherFaces and LBP, which was around 80% success. However, another important factor to evaluate the efficiency of an algorithm is the computational time required for it to perform, i.e., the time it takes until it stops. In this case, if it was a facial recognition real-time system, it was not acceptable to wait several minutes for the person to be (or not) recognized.

In Table 2.1, one can see the differences regarding the computation times on each of the three algorithms, obtained on the ORL database (see Section 3.6 for a description of this database).

Algorithm	**Training Time (ms)**	**Prediction Time (ms)**
EigenFaces	7956	8.0
FisherFaces	5883	0.7
LBP	648	40.0

Table 2.1: Computing times required by the three presented algorithms: training time represents the time required to obtain a model (which includes the full database), prediction time is the time required to perform a prediction of a subject. Results obtained using ORL database.

All of the algorithms are actually fast in terms of the recognition process (prediction process). The training process can take a lot more time, which increases proportionally to the size of the data (40 subjects in this case), because it consists on training a single model of the entire database. Although both training and prediction processes in these algorithms are different from the one presented in this thesis, they shall be compared, so one can have an idea on the differences between them and of the quality of the proposed algorithm.

CHAPTER 3

Data compression

This chapter is dedicated to the introduction of data compression, namely image compression, which is the basis of this work. Some basic notions regarding compression are discussed, as well as some motivations to use it on facial recognition. Next, the compression algorithm used on this work, the FCM, is presented and explained. There will be a short description of predictive methods, which were also subject of study in this thesis. Finally, the practical implementation of these algorithms will be explained.

3.1 Information theory

Before talking about data compression itself, one must start with the understanding of information, and relevant concepts about it. In the late 40s, Claude Shannon created the denominated Information Theory. Shannon tried to develop ways to measure the amount of information of a symbol or event without considering its meaning. He discovered the relation between the logarithmic function and information, and showed that the amount of information (or content) of an event with probability p is $-\log_b p$, also denominated the self-information associated with that same event.

As can be seen by the plot of the logarithmic function (Figure 3.1), if the probability of an event is one, the function is zero, which means that the event does not have any information; on the other hand, as the probability approaches zero, the information of that event approaches infinity.

The information unit depends on the base b of the logarithm: $b = 2$, bits; $b = e$, nats; $b = 10$, Hartleys [31].

Now, there are two concepts from this theory which are needed to understand the principles behind data compression methods: they are the concepts of entropy and redundancy.

The entropy, as defined by Shannon, is a real number proportional to the minimum yes/no questions needed to get an answer to some question. On other hand, entropy can be seen as a quantity that describes the amount of information present in a certain experiment [31].

Given a random experiment X that can have N different values, each one with a probability P_i, the average self-information of the random experiment, or the entropy, is given by Equation 3.1

$$H(X) = -\sum_{i=1}^{N} P_i \log_b P_i \ . \tag{3.1}$$

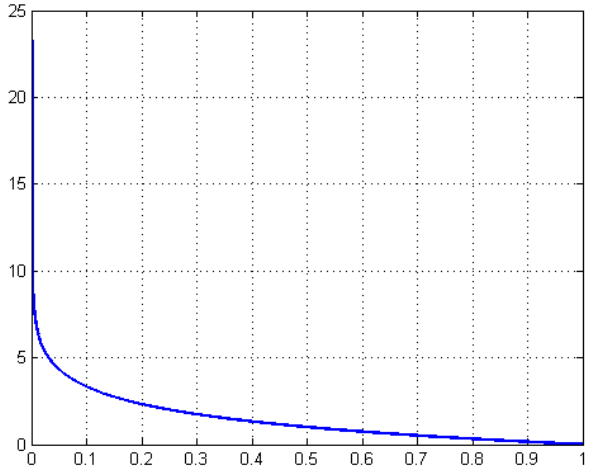


Figure 3.1: Logarithmic function $-\log_2(x), x \in [0, 1]$.

Why the usage of the logarithm on the above expression? Consider an integer n between 1 and 64. Let this integer be number 12. How many yes/no questions are needed to find this value? The answer is 6:

1. $n \leq 32$? Yes;
2. $n \leq 16$? Yes;
3. $n \leq 8$? No;
4. $9 \leq n \leq 12$? Yes;
5. $9 \leq n \leq 10$? No;
6. $n = 11$? No, so $n = 12$.

Note that 6 is the number of times 64 can be divided in half, which is equivalent to $2^6 = 64$ or $\log_2 64 = 6$; this is why logarithms are important in quantifying information.

Now, consider two symbols A and B with associated event probabilities P_A and P_B, respectively (recall that $P_A + P_B = 1$). The entropy is calculated by Equation 3.1 (consider $b = 2$, bits); let this experiment correspond to a toss of a coin (A = heads, B = tails): $P_A = P_B = 0.5$ then the entropy is 1, which means that at least 1 bit is needed to encode each symbol, which corresponds to the maximum value of entropy in this experiment, meaning the redundancy is zero, therefore the data can not be compressed.

Now, consider the case of a biased coin where the probabilities are different, i.e., $P_A = 0.8$ and $P_B = 0.2$. Calculating the entropy, a value of $H = 0.72$ is obtained: on average, each symbol can be encoded using 0.72 bit, which means that 1000 trials of this experiment, for example, could be represented approximately with 720 bit, i.e., redundancy exists on this experiment.

Redundancy is then defined by Equation 3.2:

$$R = \sum_i P_i l_i - \sum_i -P_i \log_2(P_i) \ , \tag{3.2}$$

where l_i represents a l_i-bit-long-code used to replace each of the experiment's symbols when compressing the data and $\sum_i P_i l_i$ represents the average code length (recall that $l_i = 1$ in the

experiment above) [31]. From this expression, one can conclude that the redundancy is zero when the average code length equals the entropy, i.e., the compression is maximum and the codes are the shortest.

The redundancy can also be given by Equation 3.3

$$R = \log(n) - \sum_i -P_i \log_2(P_i) \tag{3.3}$$

where n is the number of symbols.

3.2 DATA COMPRESSION

Compression is used just about everywhere. All the images one gets on the web are compressed, typically in the JPEG, PNG or GIF formats, most modems use compression, HDTV is compressed using H.264, and several file systems automatically compress files when stored.

The concept of data compression is a very old one, and since a long time ago people have used it. One example is the well known Morse-code, back on the 19-th century. In this code, letters are sent by telegraph encoded with dots and dashes. But there is no coincidence that the letters e and t are represented by a dot and a dash, respectively: they are the most frequent letters in the English language and, consequently, appear more often than others. The assignment of shorter codes to most frequent letters led to a reduction of the time required to send a message [32].

A compression method generally consists of two components: an encoding algorithm, where the input data is transformed to a compressed version, which normally has fewer bits than the original data, and a decoding algorithm, which reconstructs the original data based on the compressed one. Most of these methods are based on the same principle: removing redundancy, which, on other hand, makes the data less reliable, more prone to errors [33], as can be seen in Figure 3.2.

In this work, the data to compress are basically images, so, data compression is reduced to image compression. Some image compression aspects are presented on Figure 3.3.

One can consider different views or ways to classify image compression:

1. Information preservation - usually the way to distinguish different types of compression algorithms, based on how close the reconstructed image is to the original (often measured using some distance metric); can be distinguished between three types of algorithms:
   - Lossless algorithms, where the data can be reconstructed back to its original form from the compressed one, without loss of information. Consists on the reduction of statistical redundancy. Achieves smaller compression rates.
   - Lossy algorithms, which can only reconstruct an approximation of the original data. Consists on reducing perceptual redundancy, in other words, elimination of low relevance information that is not perceptive to human senses. Achieves larger compression rates.
   - Near-lossless algorithms, based on the reduction of statistical redundancy and a controlled reduction of the perceptual one. Intermediate compression rates.
2. Origin - the compression can be performed due to the existence of different redundancy types, namely, statistical (in which lossless algorithms are based) or perceptual/psycho-visual (lossy algorithms).

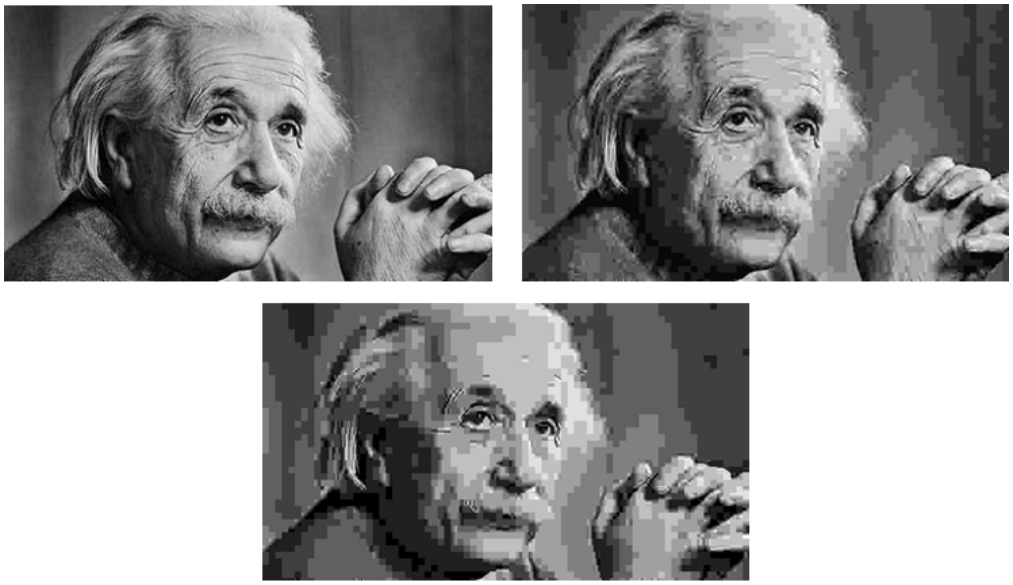


Figure 3.2: Different compression rates on a JPEG image. A greater compression ratio (right and down sub-figures) leads to loss of detail in the images.

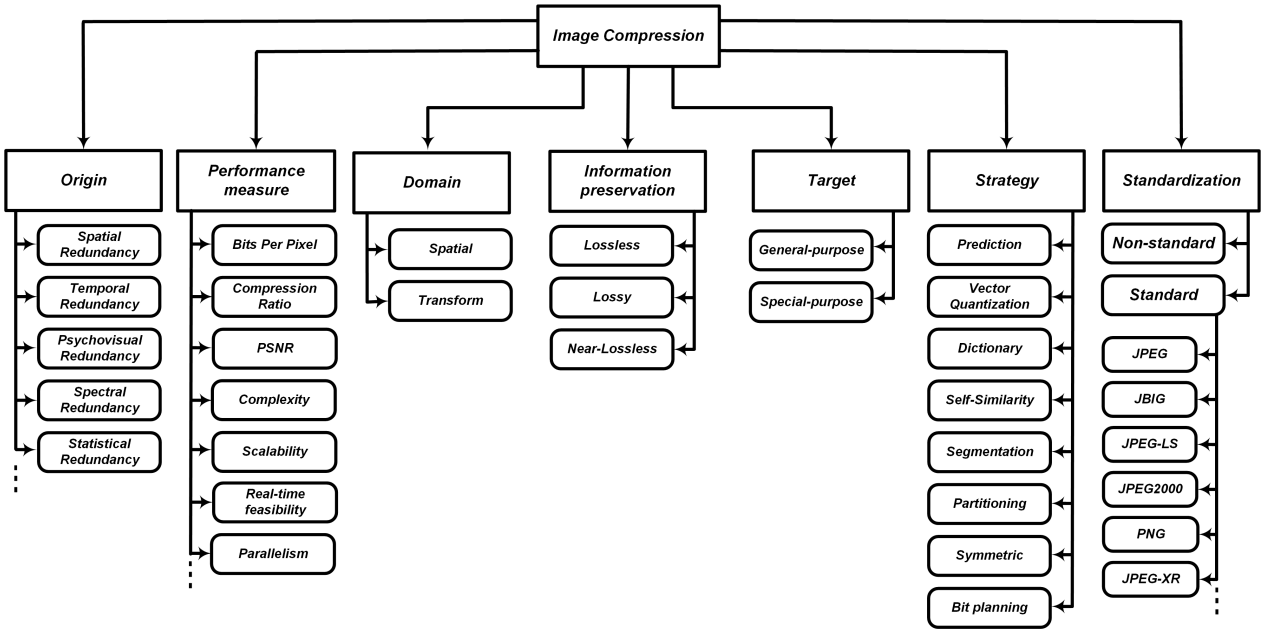


Figure 3.3: Image compression - different aspects and views [37].

3. Performance measure - set of metrics to evaluate the performance of some compression method, either by the compression ratio or bits per pixel (basically, relations of quantity of bits between original and compressed images), or even the Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) (these last two on lossy algorithms), among others.

4. Domain - compression can be performed on the pixel domain (spatial), or on the transform domain (the output is obtained through a transform function like Discrete Cosine Transform (DCT)).

5. Target - the purpose of a compression method: can be designed for general purpose or special purpose, like medical imaging, where the images have some common features.
6. Strategy - the technique used to perform compression. One of the examples are the predictive methods, which will be explored later on this chapter.
7. Standardization - compression based on standards; JPEG, JPEG-LS.

From a few years now, there has been a huge increase on the amount of information/computational data generated worldwide and, although the storage devices have really great capacity, they will eventually fill, unless one throws away some data, which is usually unwanted. Because of this fact, there is actually a big need for compression. Principles, techniques and algorithms for compressing different types of data are being developed at a fast pace by many people and are based on concepts borrowed from techniques as varied as statistics, finite-state automata, space-filling curves and Fourier and other transforms [33].

3.3 IMAGE COMPRESSION IN FACE RECOGNITION

From a real application point of view, an important, yet often neglected, part of any face recognition system is the image compression. In almost every imaginable scenario, image compression seems unavoidable. Just to name a few:

- Image is taken by some imaging device on site and needs to be transmitted to a distant server for verification/identification;
- Image is to be stored on a low-capacity chip to be used for verification/identification (we really need an image and not just some extracted features for different algorithms to be able to perform recognition);
- Thousands (or more) images are to be stored: a set of images of known persons needs to be used in comparisons when verifying/identifying someone.

All the described scenarios would benefit from the usage of compressed images, reducing the storage and transmission requirements. Compression was recognized as an important issue and is an actively researched area in other biometric approaches [35]. Usually, image compression is obtained by throwing away information that is not much relevant to the human visual system; however, there are some situations where it is necessary that the original image has to be fully reconstructed from the encoded information.

There is work of some researchers, such as Kolmogorov [28], regarding the problem of defining a complexity measure of a string. The Kolmogorov complexity of a string x, *K(x)* is defined as the length of the shortest effective binary description of x. Broadly speaking, given a string of bits x, its Kolmogorov complexity is the minimum size of a program that produces x and stops. However, this measure of Kolmogorov complexity is not computable, so it needs to be approximated by other computable measures [29], [30].

One of the choices to approximate the Kolmogorov complexity is through the usage of lossless compression, because such algorithms can be used (together with a decoder) to reconstruct the original

data, and the amount of bits required to represent both the decoder and the bit-stream can be considered an approximation of the Kolmogorov complexity [36].

This concept of Kolmogorov complexity opens the definition of information distance between objects or, in other words, similarity. For example, on [38], a similarity metric was proposed defined as the length of the shortest binary program needed to transform strings a and b into each other. This metric depends not only on the complexities of a and b but also on conditional complexities, i.e., the complexity of a when b is known.

According to [38], a compression algorithm needs to be normal in order to be used to compute this metric, in other words, the algorithm must create a model while the compression is performed, accumulating knowledge of the data (finding dependencies, collect statistics).

There has been some interest on the concept of image similarity measures based on compression methods, namely the use of compression standards like JPEG and JPEG2000, both on the spatial (pixel) and the transform (compressed) domains [3].

A Finite Context Model provides, on an "on-line" symbol by symbol basis (the model is updated while the compression is performed, i.e., as the data is processed), an information measure that corresponds, in essence, to the number of bits required to represent a symbol, conditioned by the accumulated knowledge of past symbols.

3.4 FINITE CONTEXT MODELS (FCM)

A finite-context model [34], [35] collects statistical information of a data source, retrieving probabilities of occurrence of certain events. Let it be considered an information source that generates symbols s from an alphabet $\mathcal{A} = s_1, s_2, ..., s_{\mathcal{A}}$, where $x^t = x_1 x_2 ... x_t$ represents the sequence of symbols generated after t symbols, as shown on Figure 3.4.

For every outcome, the Finite Context Model assigns probability estimates to the symbols of the alphabet $\mathcal{A}$. These estimates are calculated taking into account a conditioning context computed over a finite and fixed number k (the context size, which is a positive integer value) of the past k outcome symbols $c^t = x_{t-k+1}, ..., x_{t-1}, x_t$.

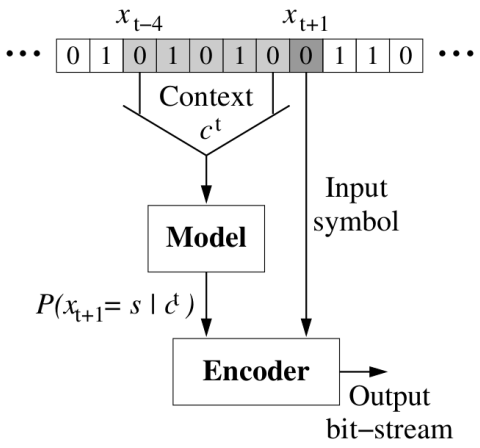
On a vectorial data input, c^t refers to the k most recently processed samples. However, on the scope of this thesis, c^t refers to the spatially closest samples (on images, the notion of recent past usually refers to spatial proximity). When the context size is small, the model explores the local statistics. On the other hand, when the window size is of the same size as the full past, the model explores the whole image statistics.

The probability that the next outcome, X_{t+1} is $s \in \mathcal{A}$ is obtained by Equation 3.4

$$P(X_{t+1} = s|c^t) = \frac{N_s^t + \alpha}{\sum_{a \in \mathcal{A}} N_a^t + |\mathcal{A}|\alpha} \ , \tag{3.4}$$

where $|\mathcal{A}|$ is the size of the alphabet, N_s^t represents the number of times that the source generated the symbol s having context c^t in the past and $\sum_{a \in \mathcal{A}} N_a^t$ is the total number of times the same context has appeared in the past.

The α parameter is used to solve probability zero problems, corresponding to the first time a certain symbol appeared after a certain context; although this event remains unseen until a certain moment in time, there's still a certain probability of occurrence of that same event. So, this parameter



Note: the encoder step on this image is not used on this thesis subject.

Figure 3.4: Example of a Finite Context Model for a binary alphabet $\mathcal{A} = \{0, 1\}$ [35].

controls (estimates) how much probability is assigned to these unseen (although possible) events. On the case of this work, this value is set to one, which is denominated as the Laplace's estimator.

The number of bits that are required to represent symbol x_{t+1} is given by $-\log_2 P(X_{t+1} = x_{(t+1)}|c^t)$, thus the entropy after encoding N symbols is given by Equation 3.5

$$H_N = -\frac{1}{N}\sum_{t=0}^{N-1} \log_2 P(X_{t+1} = x_{(t+1)}|c^t) \quad bps \ . \tag{3.5}$$

A Finite Context Model can be trained on-line (as the image is processed). In this case, a table collects counters for the number of times each symbol occurs followed by each context. As an example, suppose that at some processing instant, there is the following information (Table 3.1):

x_{n-3}	x_{n-2}	x_{n-1}	**N(0)**	**N(1)**
0	0	0	10	25
0	0	1	4	12
0	1	0	15	2
0	1	1	3	4
1	0	0	34	78
1	0	1	21	5
1	1	0	17	9
1	1	1	43	22

Table 3.1: Finite context model example.

Consider the present output is symbol "0" after the sequence "100". In this case, the probability of this output is given by $\frac{34}{34+78} \approx 0.30$, which means that $-\log_2(0.3) \approx 1.74$ *bit* are needed to encode it.

Summarily, an image is scanned pixel by pixel. If a certain pattern is found for the first time, it will require a certain number of bits to be represented, corresponding to a certain complexity (refer to Kolmogorov complexity). When that same pattern is seen again, the number of bits needed to encode that occurrence will be smaller, because the model has constructed an internal representation

of the pattern. This leads to the following: when a Finite Context Model is trained with a certain image, if one tries to encode the same image based on the trained model, the total number of bits will be smaller than the one to encode the image with no model. Plus, if the same model is used to process two different images, the number of bits will be smaller in the case of the most similar image (compared to the model) (Figure 3.5).

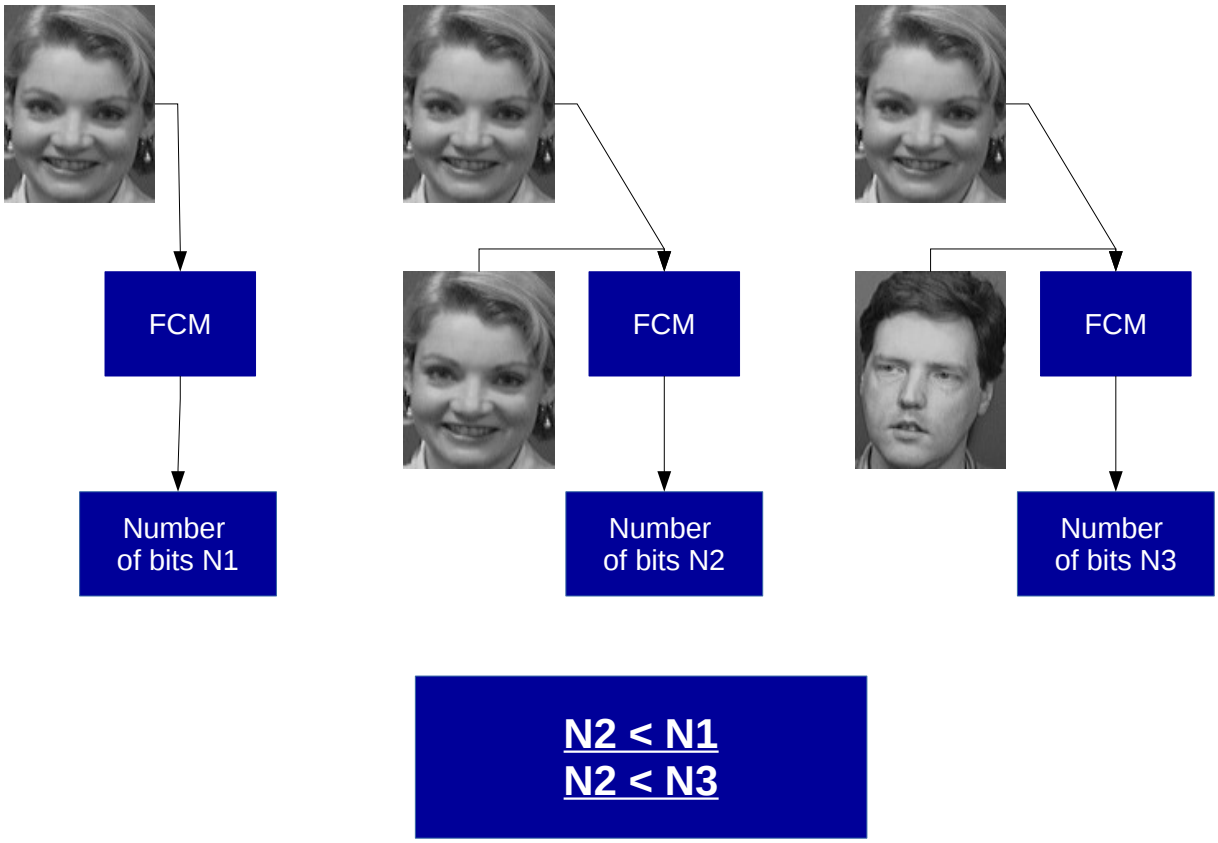


Figure 3.5: Principles of FCM on facial recognition.

Summarily, if on a database exists a certain test image of a subject, it is expected that the minimum number of bits needed to encode it is obtained with a previously trained model of the same subject (because it is expected to be the most similar between all the subject's models).

On the subject of this work, there are some points to note:

- The context can be selected to vary between 1 and 8, corresponding to the closest neighbors of a certain pixel (Figure 3.6). This value is limited because of memory and time resources; the pixels chosen are the spatially closest, because they are believed to correspond to the most recent past (in terms of statistics, they are more related with the central pixel than those further away on the image).
- The alphabet depends on the number of quantization levels of the images (which will be introduced in Chapter 4), but its maximum size is 256, which corresponds to the full range of intensities of gray-scale 8-bit images.

- The term of comparison is the total amount of bits B needed to encode a certain image, which is the sum of the number of bits to represent each pixel of the image (Equation 3.6)

$$B = -\sum_{t=0}^{N-1} \log_2 P(X_{t+1} = x_{(t+1)}|c^t) \quad bit \quad . \tag{3.6}$$

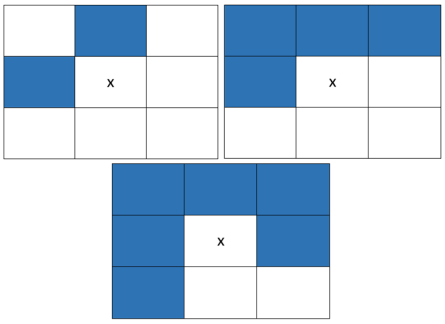


Figure 3.6: Examples of the context used on the Finite Context Model; 2, 4 and 6 pixels respectively.

3.5 PREDICTIVE METHODS

In this work, there was still a study on the use of another compression approach, namely predictive methods, for facial recognition purposes. Predictive methods on images use spatial information to predict outcomes. The goal is to use the values of the closest neighbors of a pixel to predict its value.

Adjacent pixels in an image tend to be correlated, which means the difference between a pixel and any of its neighbors tends to be a small integer. Because the value from a pixel may turn to be very different from its neighbors, some more sophisticated predictors compare the pixel with an average of its nearest neighbors. The difference between original and predicted pixels, i.e., the prediction error, tends to be distributed following a Laplacian distribution.

If the prediction is done properly, the prediction error tends to be small, which means the pixels were correlated. On this case, they are easy to compress by replacing them with variable-length codes. [31].

This step (prediction, calculation of the error and subtraction) is also known as decorrelation, because one decorrelates the data by subtracting the best possible prediction of it. This decorrelation is in fact the usual interpretation of the beneficial effect of prediction, as it allows the use of simpler models for the prediction errors [41], [42].

Figure 3.7 shows some pixels which can be used to predict some other pixel's value (pixel s in this case).

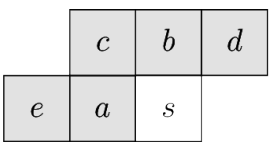


Figure 3.7: Spatial information (pixels) used to predict value of current pixel s [47].

The type of prediction used on this work is the same as in the JPEG-LS standard, know as the LOCO-I algorithm [40], where the prediction of the current pixel s is performed as follows (Equation 3.7):

$$s = \begin{cases} \min(a,b) & \text{if } c \geq \max(a,b) \\ \max(a,b) & \text{if } c \leq \min(a,b) \\ a+b-c & \text{otherwise} \end{cases} \tag{3.7}$$

Next, the prediction error is calculated, subtracting the predicted image from the original one. An example of this error can be seen in Figure 3.8.

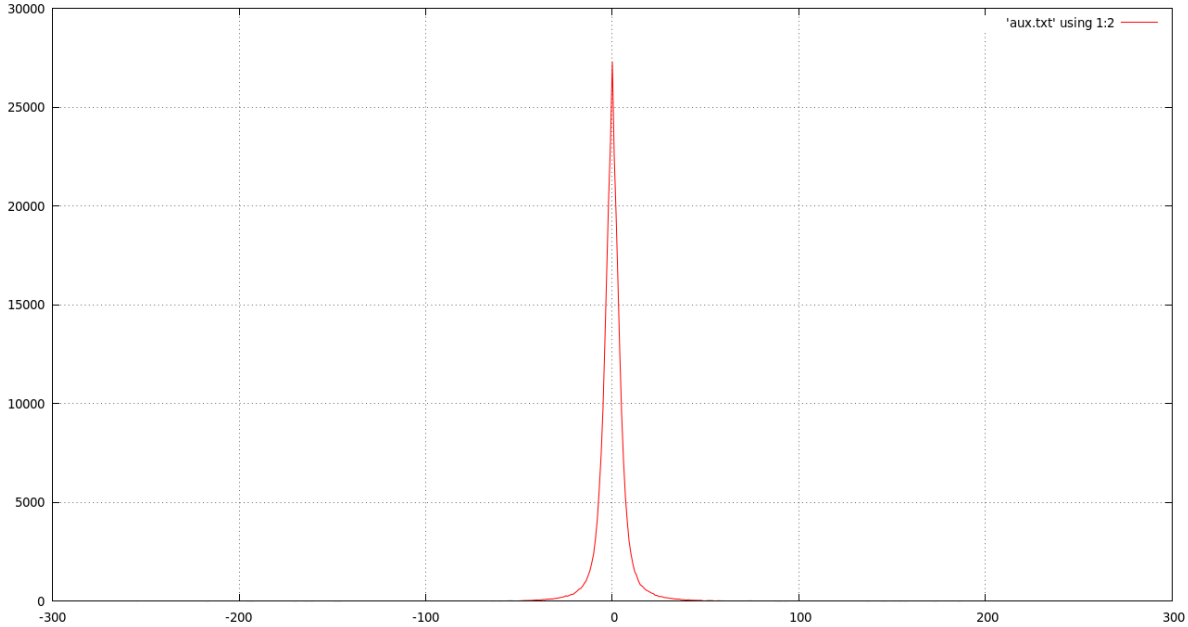


Figure 3.8: Prediction error of 'lena.pgm'(512x512 pixels) using LOCO-I prediction algorithm.

In the plot, the x-axis represents the error, which is a discrete value $i \in [-N+1, N-1]$, where N is the number of intensities of the image (256 in this case), because that is the worst scenario possible (some pixel presents a value of 255 in the original image and 0 on the predicted one); the y-axis represents the number of times that error was calculated.

The entropy can be calculated by Equation 3.8

$$H = -\sum_{i=-N+1}^{N-1} P(\epsilon = i) \log_2 P(\epsilon = i) \quad bps \ , \tag{3.8}$$

where $P(\epsilon = i)$ represents the probability of a certain error, i.e., the number of times that error appeared over the total number of pixels.

This value of entropy can be used as a comparison between images. However, for this purpose, it is easier to compare them using a simpler metric, like the number of bits, either for a certain symbol or for the total image. So, this number can be computed as the image is processed; at a certain time, the number of bits to encode the actual symbol is given by Equation 3.9

$$m = -\log_2 P(\epsilon = i) \quad bit \ . \tag{3.9}$$

The total number of bits to encode an image can be obtained by the sum of all those bits.

The idea to use this method on facial recognition was to compute the prediction error and calculate the number of bits to encode a certain image, with a previously used image as training, meaning that, if one considers the prediction error displayed on a table of number of occurrences, that table was already "filled" with the prediction error from the first image.

Some tests were performed with the ORL database, but the results were not as good as the ones with the Finite Context Models. However, they will be presented in the next chapter.

3.6 PRACTICAL IMPLEMENTATION

Initially, a project in the C++ language was started, which implements the Finite Context Models (FCM) algorithm. In this project, a class was created: the fcm class that contains all the variables and methods necessary for a proper implementation of the algorithm described, and it is used as a basis to the executable programs. The OpenCV library was used, which features a number of useful tools to the project, including the Mat class (matrix) as well as several image processing modules. There are two files that generate executables which are the train.cpp and main.cpp files.

The first was used to, given a training subject (with multiple images) and a context size, train a model. This model consists of a .xml file with all the information on the trained subject and relevant for further recognition. The amount of bits needed to encode each of the training images using the trained model is estimated, thus yielding a lower and upper limit to the number of bits for each subject. The second file, generates an executable that, with a trained model and an image of a subject passed as arguments, calculates the number of bits required to encode the image using that model.

For compiling, the Linux compiler, gcc, present in Ubuntu (version 14.04) was used, the operating system used for this project. For a simple and versatile implementation, CMake was used by creating CMakeLists.txt configuration files.

Regarding the databases for the following tests, two were used, as already mentioned: the ORL face database and the FERET database. Both cases are composed by 40 test subjects (people) where images are in the .pgm format, 8 bits per pixel gray-scale, with size 92x112. In the first case (ORL), 9 images are used to train the model and a different one to test. These pictures were taken at different times, with different conditions: lighting variations, expressions and facial details. All the images have a black background and the subjects are in a forward position relative to the camera with a small margin for lateral rotation. At the 2nd (FERET), only three training images were used (front, left side and right side) plus a frontal image as test. Due to the fact that these 3 images are very different between them, the results are predicted in the case of this database. Examples of the faces can be seen in Figure 3.9.

Figure 3.9: Example of the images used in the tests. ORL Database above, FERET below.

The class which implements the Finite Context Models can be reviewed in Listing 1 .

```
class fcm
{
  uint8_t *context;
  bool two_dim_context;
  Mat fcm_table;
  Mat sum_table;
  uint16_t n_values;

  public:
    fcm();
    fcm(uint8_t n);
    fcm(uint8_t n, vector<Point> pixel_rel_pos);
    ~fcm () { delete context; fcm_table.release(); sum_table.release();}
    uint8_t context_size;
    void print_table_until(int row, int col);
    void update_table(Mat m,uint8_t px, uint8_t py, vector<Point>
        pixel_rel_pos);
    double n_bits(uint8_t next_symbol, double sigma);
    void reset_table();
    void save_model(const char *c, double max_bits, double min_bits);
    void load_model(const char *c, double &max_bits, double &min_bits);

  private:
    void init_fcm_table(void);
    void get_context(Mat m,uint8_t px, uint8_t py, vector<Point>
        pixel_rel_pos);
    void print_context(void);
    uint32_t get_row(void);
};
```

Listing 1: The FCM class header file.

The FCM class, as any other C++ class, has its own members: some variables needed to implement

the described algorithm as well as some methods which work with those variables. Starting with the variables, there is only a public one, because it was needed in the main program; still, all the other ones are private, because they are not needed to be accessed from the outside, except for the class's own methods. The created variables are the following:

- `context` - an array whose members are the symbols used as context, i.e., the intensities of the pixels used as context.
- `two_dim_context` - a flag that signalizes if the context is one/two directional, i.e., if the pixels used as context are the previous ones on the same row, or are spatially distributed around the pixel being processed.
- `fcm_table` - the most important variable of the class; a matrix that represents the table as shown in Table 3.1, which contains the counters for each context, for each image intensity.
- `sum_table` - matrix that contains the number of times the context has appeared, basically represents the row-by-row sum of the previous `fcm_table`, but it is useful to a faster computation of the probabilities.
- `n_values` - the size of the alphabet used, i.e., the number of different intensities of the image.
- `context_size` - represents the size of the context, i.e., the number of pixels to be used as context; it is public because it can be a parameter of the main program for the user to input.

Regarding the methods, there are some public and private as well; on the public members there are the following:

- `fcm()`, `fcm(n)`, `fcm(n,pixel_rel_pos)` - overloaded constructors to allow the user to choose between the default configuration (the context are the two previously horizontal pixels, by previously consider the left ones), to insert the size `n` of the context (but still one directional) or to insert the size to use on a two directional context (`pixel_rel_pos` represents the Cartesian coordinates of the pixels to use as context, considering the actual processing pixel as the origin of the referential).
- `~fcm()` - class's destructor, followed by the deletion of the tables from the computer memory.
- `print_table_until(row, col)` - prints the actual state of the table, i.e., the values of the counters, until the row and column passed as parameters (if row = 0 and col = 0 prints the full table); it is used for debugging, to check if the counters were correctly being updated.
- `update_table(m, px, py, pixel_rel_pos)` - method used to update the counter of the table, i.e., it searches the table for the actual pixel intensity and the correct context and increments the respective counter by one. Parameter `m` is the input image, `px, py` the position of the pixel being processed, and `pixel_rel_pos` is the same as on the "two-dimensional constructor".
- `n_bits(next_symbol, sigma)` - method used to retrieve the number of bits to encode a certain symbol (obtained by variable `next_symbol`). `sigma` represents the α parameter on the probability formula of the Finite Context Models. It takes value one, i.e., it is the Laplace estimator.
- `reset_table()` - resets the counters on the fcm and sum tables.

- `save_model(c, max_bits, min_bits)` - saves the current state of the fcm, creating a model, i.e., a *xml* file `c` containing relevant information on the fcm, with a range of acceptance of a test image given by the values of `min_bits` and `max_bits` which are local variables.
- `load_model(c, max_bits, min_bits)` - loads the model `c` to an object of the type fcm.

On the private members, there are the following:

- `init_fcm_table()` - sets the initial state of the fcm variables, depending on the type of constructor called.
- `get_context(m, px, py, pixel_rel_pos)` - loads, to the `context` variable, the context of the actual pixel being processed `(px,py)` on the image `m`.
- `print_context()` - method used to print the context of the actual pixel; used for debugging to check if the context is being retrieved correctly.
- `get_row()` - returns the number of the row on the fcm table to which corresponds the actual context; it is used to update this same table.

On other hand, a predictor class was implemented to try to use predictive methods on face recognition. The header file of this class can be seen on Listing 2 .

```
class predictor
{

  public:
    predictor();
    predictor(uint8_t n);
    ~predictor () { pred_table.release(); aux_img.release();}
    Mat pred_table;
    double n_bits;

    void calc_aux_img(Mat m);
    void calc_pred_table(Mat &m);
    double get_entropy();
    double get_bits(int16_t error, uint16_t row);
    void print_pred_table();
    void print_aux_img();
    void reset_table();

  private:
    Mat aux_img;
    uint16_t n_values;
    uint8_t select_value;
    int16_t get_prediction(Mat &m, uint16_t col, uint16_t row);

};
```

Listing 2: The predictor class header file.

Once again, on this class there are some variables and methods, which will be briefly presented; on the variables:

- `pred_table` - a matrix with 2 columns, where the first one represents the prediction error (from $-(N-1)$ to $(N-1)$ where N is the number of intensities of the images) and the second one corresponds to the number of times that value of the prediction error was found.
- `n_bits` - number of bits needed to encode a certain pixel.
- `aux_img` - resulting image from the application of the prediction scheme on the original image.
- `n_values` - number of intensities of the images.
- `select_value` - value to select the prediction scheme which will be used to perform the algorithm; can go from 0 to 8, each value selects a different prediction scheme (8 corresponds to the LOCO-I algorithm).

There are some public methods on this class to perform on the variables; they are:

- `predictor(), predictor(n)` - overloaded constructors to allow the user to choose between the default configuration (which is 8, corresponding to the LOCO-I algorithm) or to select another prediction scheme n, from 0 to 8.
- `~predictor()` - class's destructor, followed by the deletion of the tables from the computer memory.
- `calc_aux_img(m)` - method which calculates the auxiliary image, applying the prediction scheme on the original image.
- `calc_pred_table(&m)` - method used to calculate the prediction error table. This method processes the image and for each pixel calculates the prediction error and increments its value on the prediction table.
- `get_entropy()` - returns the entropy of the information source (the image), based on the prediction error counters.
- `get_bits(error, row)` - returns the number of bits needed to encode the pixel being processed, based on the prediction error of that same pixel and its row on the prediction error table.
- `print_pred_table()` - prints the prediction error table, used for debugging.
- `print_aux_img()` - debug method, prints the auxiliary image to check if it is being correctly generated.
- `reset_table()` - resets the prediction table to its initial state.

There was created only one private method:

- `get_prediction(m, col, row)` - method used to get the predicted value of a certain pixel (col, row) on image m, based on the previously chosen prediction scheme (`select_value`).

3.7 FINAL REMARKS

In this chapter, the theme of data compression was introduced, and the concepts needed to understand the principles behind the compression algorithms were explained.

These algorithms, which were presented and explained, were then both connected to the facial recognition problem; this connection allows to answer the main question of this thesis about the use of these compression algorithms to recognize faces.

The implementation of these algorithms were presented, so one can have a practical idea of how the results shown on the next chapter were obtained.

CHAPTER 4

Face Recognition using Finite Context Models

This chapter is dedicated to the experimental evaluation of the previously referred compression algorithm (Finite Context Models) on facial recognition. First, this algorithm was implemented on a C++ based project with the objective to study and test its effectiveness. Two face databases were used as test subjects: "The Database of faces"(formerly the ORL Database of faces), and the Face Recognition Technology (FERET) database. Several tests were performed, based on the face recognition problem (number of subjects successfully/unsuccessfully recognized), with and without changes on the images conditions. Some real-life tests were still performed. The chapter ends with a small presentation of the recognition results using predictive methods.

4.1 recognition tests

The first tests were basic facial recognition tests, which means, in this case, check, not only the label (or name) of the subject, but also his presence on the database. Due to the computational complexity of the algorithm, it is not practical to use the original images, since the necessary resources in terms of memory required, as well as time, follow an exponential function of type $\mathcal{A}^c$ where

- $\mathcal{A}$ is the alphabet size (number of different intensities);
- c is the model's context size.

Because of this, the images used are quantized using the Lloyd-Max quantization, which uses an algorithm for calculating the quantization levels based on the probability density function (PDF) of the pixel's intensities, in order to minimize the mean squared error (MSE) and thus maximize the SNR. The images were then followed by a re-indexing of the intensities, in order to "pull" the histogram to the left side, thus obtaining an image with intensities ranging between 0 and $(N-1)$, where N is the number of quantization levels.

All models were created for each subject and the numbers of bits required to encode each of the test images with all trained models were calculated (using the FCM algorithm). Therefore, and assuming

that the minimum number of bits to encode an image of a subject is obtained with the trained model of the same subject, one can draw conclusions about the success of the face recognition. Thus, three different situations may occur:

- The minimum number of bits to encode the test image is obtained with the trained model of the same subject, and this number is within the limits (in terms of numbers of bits) of this model - subject successfully recognized;
- The minimum number of bits to encode the test image is not obtained with the trained model of the same subject, but this number is within the limits - subject recognized incorrectly (false positive);
- The minimum number of bits to encode the test image is outside the limits of the model - subject not recognized.

Below, 4 examples of plots are given (Figures 4.1, 4.2, 4.3, 4.4) (obtained resorting to the Octave Software) of the results performed for both databases, using quantized images with 8 levels and a context of 2 and 4 pixels (all of the plots of the results can be consulted on Appendix A of the thesis).

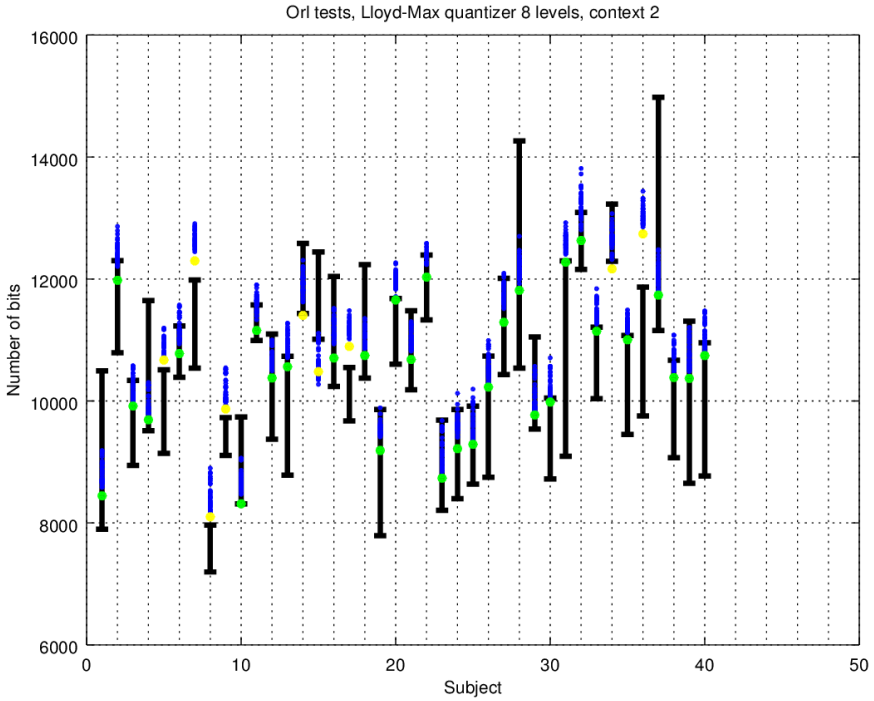


Figure 4.1: Recognition results using the ORL database (context 2).

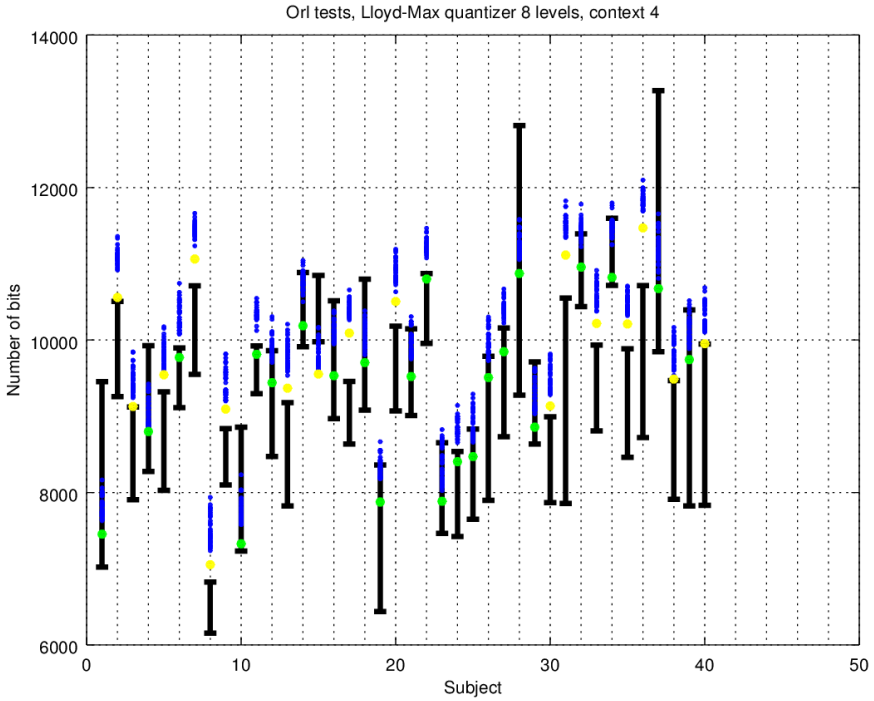


Figure 4.2: Recognition results using the ORL database (context 4).

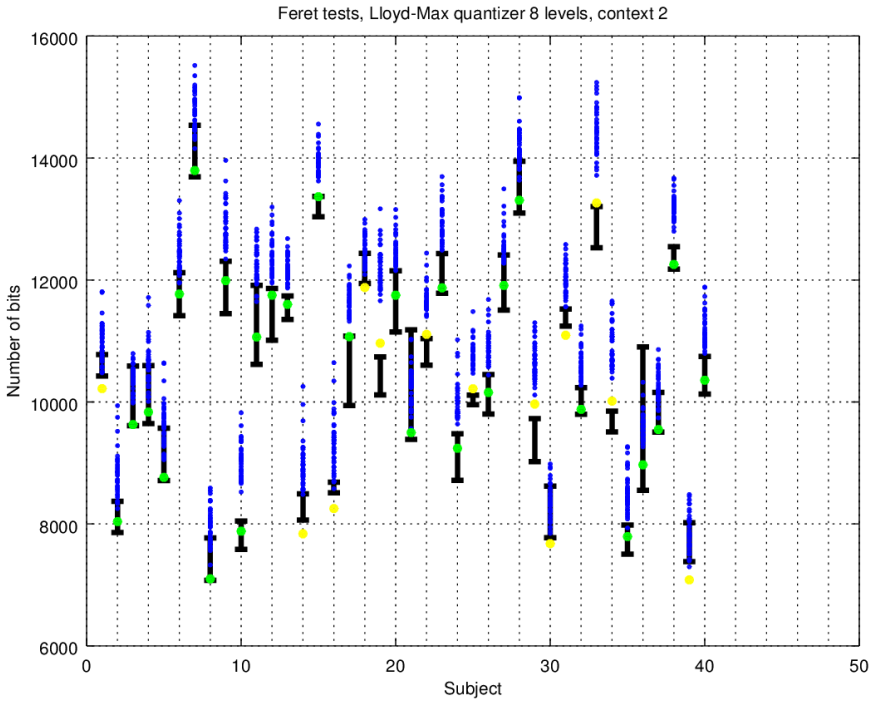


Figure 4.3: Recognition results using the FERET database (context 2).

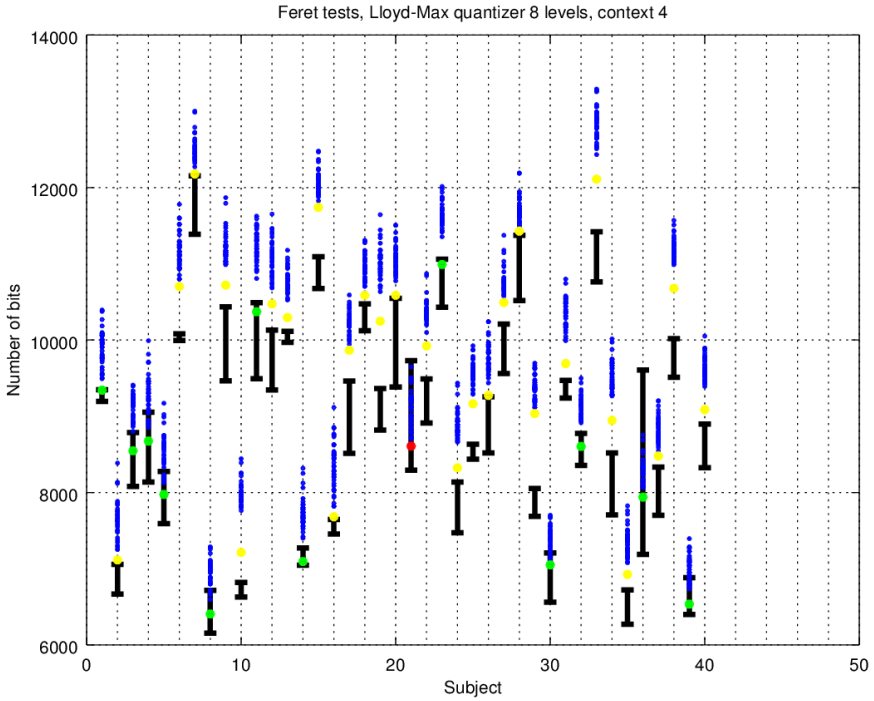


Figure 4.4: Recognition results using the FERET database (context 4).

In these plots, for each subject's test image (horizontal axis), one can observe the upper and lower limits of bits of the respective model to accept an image (range of black values) as well as the number of bits to encode the image with each one of the 40 models (other colors).

The blue points represent the number of bits to encode each of the test images with each model that is not from the same subject. The points that are not highlighted in blue represent the minimum for each subject: green, the test image corresponds to the same training subject (positive); red, the test image does not belong to the same subject (false positive); yellow, the image is out of the model's acceptance limits (no recognition).

It can be seen that there are subjects with a wide range of acceptance (for example the model 28 from the ORL database), which can withstand some variations in the test image, but can also lead to wrong recognition of different test images from other subjects; at the other end there is, for example, model 16 from the FERET database, whose range is very limited, and thus only very similar test images, compared to the training set, will be accepted.

It was decided to perform the same procedure using several quantization levels and different context configurations, in order to study the effect of these on the recognition rate. This information can then be found in Tables 4.1 and 4.3 for the two databases as well as some information on the computing time (Tables 4.2 and 4.4) required in each case.

Through the observation of the Tables 4.1 and 4.3, one can draw some conclusions, namely the effect of increasing the context: the number of non-recognized faces, that is, the ones which quantity of encoding bits is not in the range of the model, increases significantly in some cases. The increase of the context results in higher compression factors (comparing plots for the same database, one realizes that the required number of bits is less in the context 4 case than in context 2) because the number of possible pixel combinations increases exponentially, which leads to lower probabilities of

Number of Quantization Levels	Context	Positives	False Positives	Unrecognized
4	2	29	5	6
	4	29	4	7
	6	31	0	9
8	2	31	0	9
	4	23	0	17
16	2	25	3	12
	3	24	1	15

Table 4.1: Recognition results using several quantizations and contexts, ORL Database.

Number of Quantization Levels	Context	Training Time Average (ms)	Recognition time Average (ms)
4	2	114	500
	4	345	1680
	6	5000	25000
8	2	125	530
	4	1900	8000
16	2	130	550
	3	780	3530

Table 4.2: Computation time (on average) to train a model and to perform the recognition of a subject on the ORL database.

Number of Quantization Levels	Context	Positives	False Positives	Unrecognized
4	2	22	4	14
	4	21	2	17
	6	20	1	19
8	2	27	0	13
	4	12	1	27
16	2	20	0	20
	3	7	0	33

Table 4.3: Recognition results using several quantizations and contexts, FERET Database.

these combinations and, therefore, a reduction of the amount of statistical information; the images are more difficult to "recognize" for the algorithm.

On the other hand, on the ORL database, since the images are very similar, using a small number of intensities to represent them (namely 4, in this case) creates regions with similar information (patterns). Here, one can see a different consequence of the context increase: when using larger contexts, the algorithm behaves like a "dictionary", making it easier to discover these patterns in a more efficient way, which leads to an increase of the Positives.

Number of Quantization Levels	Context	Training Time Average (ms)	Recognition time Average (ms)
4	2	40	460
	4	150	1180
	6	1500	12947
8	2	46	550
	4	870	5912
16	2	54	584
	3	334	2910

Table 4.4: Computation time (on average) to train a model and to perform the recognition of a subject on the FERET database.

Because the Lloyd-Max quantization is not uniform (Figure 4.5) (the quantization step is not constant in terms of the image histogram), it was decided to study if this could affect the recognition, because, although the quantized image intensities were re-indexed, the quantization levels (using Lloyd-Max) are different from image to image.

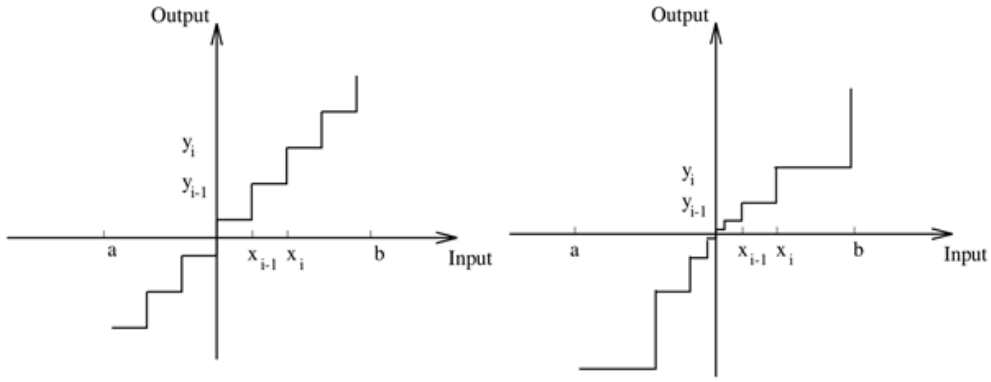


Figure 4.5: Uniform quantization on the left, non-uniform quantization on the right (Lloyd-Max case). The quantization step is constant on the first one, which is not the case on the second.

Thus, two kinds of uniform quantization, in terms of the image histogram (Figure 4.6), were implemented. In the first one, the quantization levels are chosen dividing the total possible range of pixels intensities (0 to 255 in this case) into n equal parts, where n represents the number of quantization levels; in the second case, it is not used the full range of possible intensities, but only the range of intensities present in the image, and again dividing the histogram into n equal parts. The results obtained for the FERET database with 8 quantization levels can be found in Table 4.5.

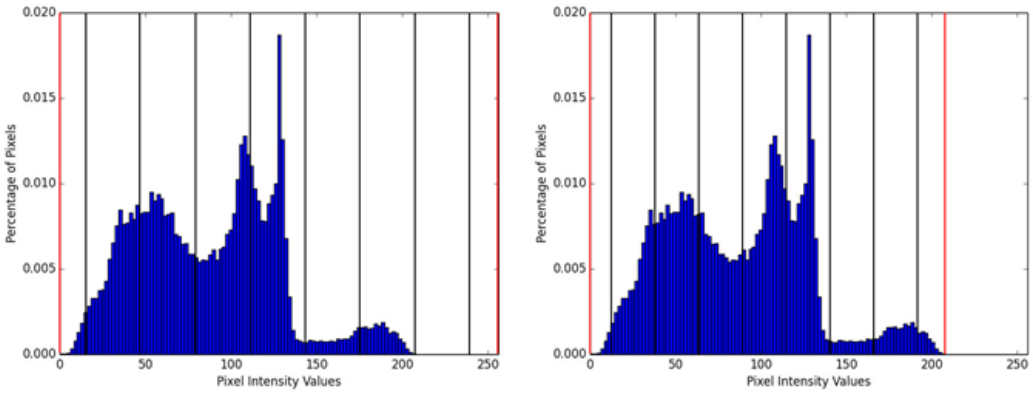


Figure 4.6: Two types of uniform quantization implemented.

Quantization	Context	Positives	False Positives	Unrecognized
Lloyd-Max	2	27	0	13
	4	12	1	27
Uniform 1	2	21	1	18
	4	17	0	23
Uniform 2	2	25	3	12
	4	16	0	24

Table 4.5: Recognition results with different types of quantization using the FERET Database, quantized with 8 levels.

Analyzing Table 4.5, one can not say that a particular quantization has advantage over the others: the Lloyd-Max quantization performed better with a context of 2, contrasting with the best behavior of uniform quantization with context 4.

Furthermore, and returning back to the plots of Figures 4.1 and 4.2, and in particular for the unrecognized subjects (yellow), two situations may occur, corresponding to the fact that the required number of encoding bits of an image to be very close or be distant from the limits of the model for it to be given as recognized. In this sense, and in order to differentiate these situations, a tolerance to the limits of the models (threshold) was added, which corresponds to a certain percentage of the minimum and maximum number of bits accepted by the model; this way, images whose quantity of encoding bits are very close, yet outside, the limits, will be given as recognize and distant images remain as unrecognized. Therefore, some tests were made to get a reasonable value for this threshold, which will be referred to as "tolerance".

On Figures 4.7, 4.8, 4.9 and Table 4.6 one can refer to this information in the case of ORL database, Lloyd-Max quantization with 8 levels, context 4.

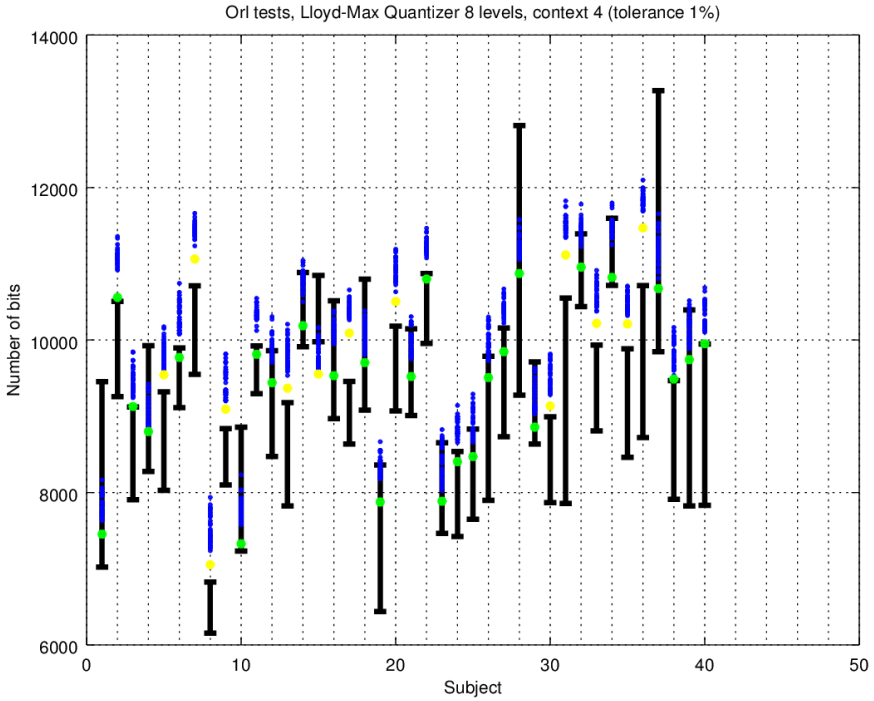


Figure 4.7: Recognition results using the ORL database quantized with 8 levels, context 4, and tolerance value of 1%.

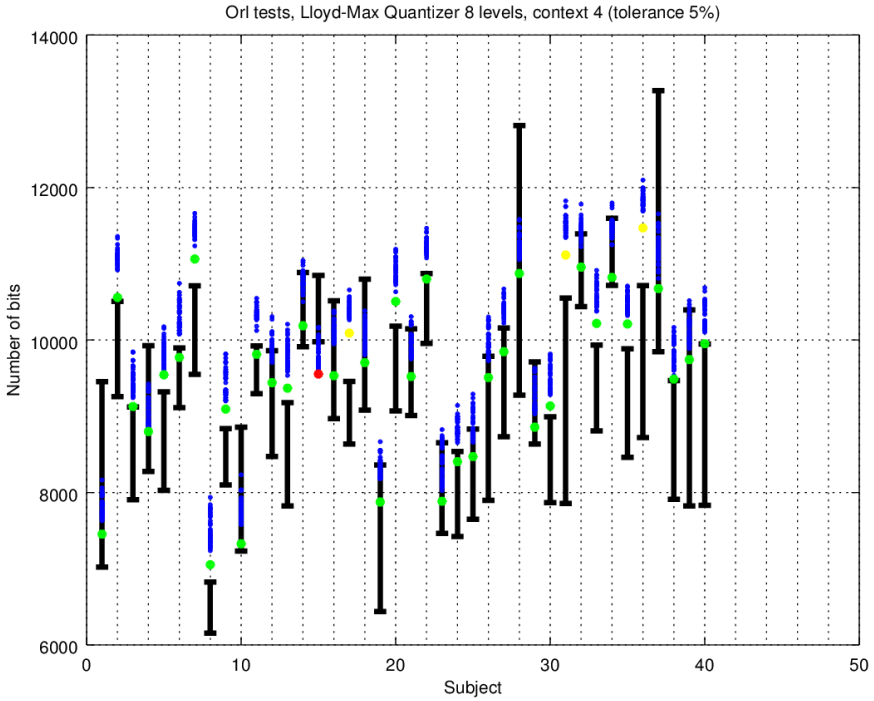


Figure 4.8: Recognition results using the ORL database quantized with 8 levels, context 4, and tolerance value of 5%.

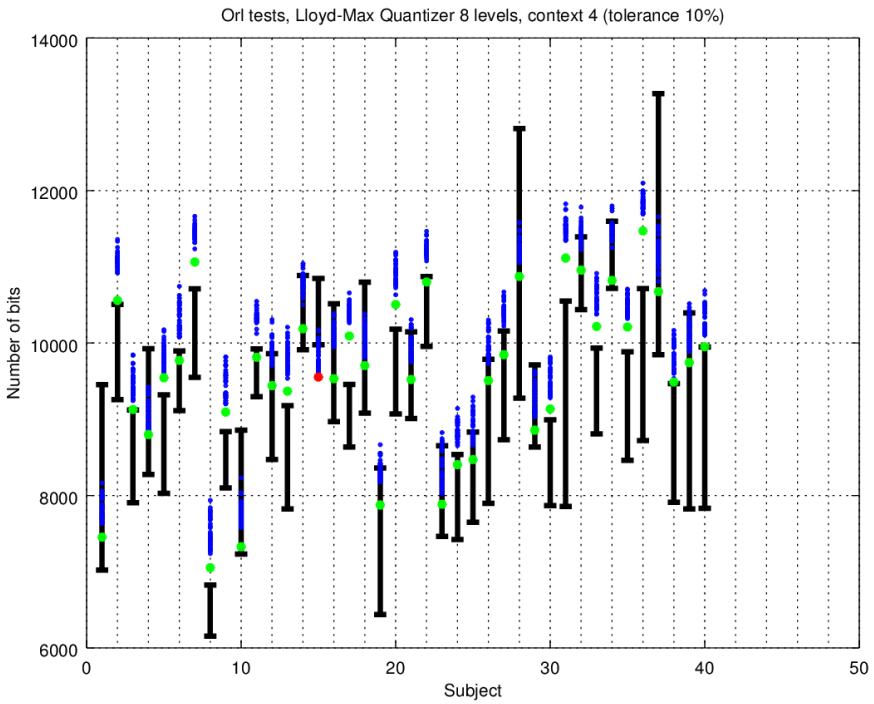


Figure 4.9: Recognition results using the ORL database quantized with 8 levels, context 4, and tolerance value of 10%.

Tolerance(%)	Positives	False Positives	Unrecognized
0	23	0	17
1	27	0	13
5	36	1	3
10	39	1	0

Table 4.6: Recognition results using the ORL database quantized with 8 levels, context 4, and tolerance value of 1%, 5% and 10%, respectively.

Observing the plots, one can conclude that it is reasonable to add a tolerance of between 1 to 5% to the decision process of recognize/not recognize a subject. A higher value would allow acceptance of a large number of images (including different subjects) which would lead to a larger confusion between classes and a reduction of the ability of the system to assign the condition of non-recognized to a particular image.

4.2 TESTS OF INVARIANCE TO CHANGES ON THE IMAGES ACQUISITION

In addition to the primary recognition tests, three more were conducted considering changes in the image conditions, in order to study the versatility of the compression algorithm in those cases.

These tests were achieved through light, rotation and scale changes on the test image. The ORL database was used, which has the better trained models, because it was the one with better results. The Lloyd-Max 8 levels quantization, context 4 was the "setup" used.

The first executed tests were the lighting ones, which is quite an important question on an outdoor facial recognition system (the train images could have been taken on a day with different weather conditions), because the recognition can change completely. In order to study such effect on the recognition by the FCM algorithm, the test images were modified multiplying each one of its pixels by a factor (Figure 4.10); this factor, or lighting factor, when less than 1 represents a reduction on the quantity of light (darker images) and, when greater than 1, it represents an increase (brighter images).

Figure 4.10: Examples of test images from the ORL Database, with lighting changes (before the quantization). Lighting factor from left to right: 0.5; 0.8; 1; 1.2; 1.5.

In Table 4.7, one can check the results of the lighting changes on the recognition process (the respective plots can be checked on the Appendix A).

Lighting Factor	**Positives**	**False Positives**	**Unrecognized**
1	23	0	17
0.5	22	0	18
0.8	25	0	15
1.2	27	0	13
1.5	14	2	24

Table 4.7: Recognition results applying lighting changes on the test images.

The conclusion drawn from these results is that this algorithm is viable to use with small changes in the lighting conditions, although the good results, even in the 0.5 lighting factor case (half of the light quantity). The increase of the positives in this case of small changes is quite a curious fact, but can be explained by some of the already present changes in the train images. In the beginning of this chapter, when the two databases were presented, it was referred that the ORL database presents some lighting variations between images of the same subject. Now, the lighting change on some test images led to the increase of similarity between these last and the train ones, resulting in some of the test images changed from a non-recognized to recognized state. The preprocessing of the images, i.e., quantization and re-indexing, might have contribute to the non-deterioration of these results.

The second test was the rotation one. The test image rotation is a relevant problem in the face recognition, because it is not desirable that, due to small image rotations, a subject becomes unrecognizable. In order to study this problem, the test images were rotated sideways by some degrees, clockwise and counter-clockwise; an example of these images may be seen in Figure 4.11, which were modified with resource to the OpenCV library. The effect of the rotation on the test images is present on Table 4.8.

Figure 4.11: Examples of test images from the ORL Database, with rotation changes (pre-quantization). From left to right: Original image, clockwise rotation (5º, 10º, 15º), counter-clockwise rotation (-5º, -10º, -15º).

Rotation (º)	**Positives**	**False Positives**	**Unrecognized**
0	23	0	17
5	8	0	32
10	7	1	32
15	5	5	30
-5	8	1	31
-10	11	2	27
-15	15	2	23

Table 4.8: Recognition results applying rotation changes on the test images.

Through the observation of Table 4.8, it can be concluded that the FCM algorithm did not perform well with image rotations in the case of this database, cause the number of unrecognized increases a lot comparing with the previous cases. Still, the number of positives doubles when using a tolerance of 5% (in the worst cases). Another curious result is the increase of the positives in the counter-clockwise case (negative degrees), which can be explained similarly to the lighting case: some training images are already rotated, so when the test image is as well rotated, it approximates the training images and the amount of encoding bits fall into the model's acceptance interval.

Finally, it was tried to execute some tests regarding the scale of the images. In a real application, the test image might be taken with resource to a facial detection algorithm, therefore the size of this image may vary. Although this is relevant, the algorithm studied on this work does not perform well with big differences on the test image size because the total number of bits to encode an image varies proportionally to its size, making it hard to compare with a training model with another size. So, when tried to use a test image with different size of the training set, the total amount of bits was not of the same order of the acceptance interval of that model (if the size changes significantly), as can be seen on Figure 4.12.

As can be seen in the plot (Figure 4.12), the total number of bits to encode the images decreases because the size decreased too, therefore all of the subjects are outside the model's limits, which leads to being labeled as unrecognized.

A possible solution is to normalize the number of bits to the size of the images, still the results obtained were a bit disappointing.

However, it is possible to work around this problem because, as the test image is usually obtained by some facial detector, it can be preprocessed before the performance of the recognition algorithm (already is, because of the quantization), so it can can be easily resized to match the size of the training set images.

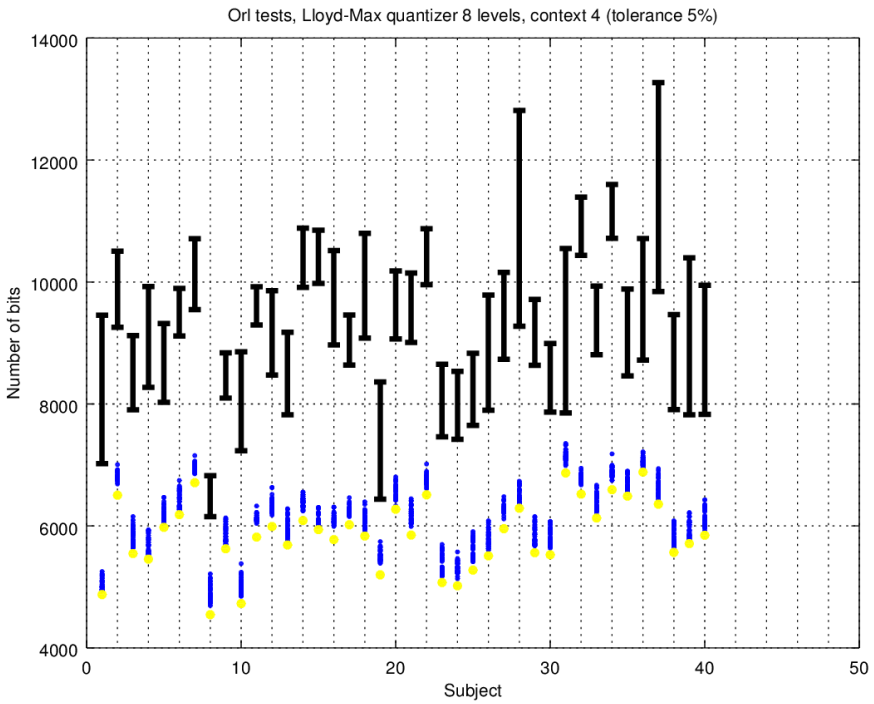


Figure 4.12: Recognition results when using a 69x93 image (smaller) with the original training set.

4.3 REAL-LIFE APPLICATION OF A FCM-FACE RECOGNITION SYSTEM

Besides the tests already presented with well known databases, there was an opportunity to implement a practical facial recognition system.

This system consisted on the following:

1. Images were taken from a Kinect camera connected to the computer; on these images, the *Viola and Jones* algorithm for face detection was applied and the faces were taken and stored (five training images were taken per subject plus a different one for testing), creating a new database;
2. The images were preprocessed, i.e., all resized to 92x112 pixels, quantized with 8 levels;
3. For each subject, the training models were created;
4. The recognition was performed.

On Figure 4.13 there is an example of a subject present in the created database.

The results presented were obtained by calculating the number of bits to encode each of the test images with each of the models (same as with the ORL and FERET databases). It was decided to use quantized 8-level images and a context of 2 because the computational time is reasonably low. Also, it was the best "combination" on the previous tests. Figure 4.14 and Table 4.9 show these results using the created database as testing (total of 27 subjects).

Once again, the Finite Context Models approach proved to work under constrained environments as can be shown by these results.

Figure 4.13: Example of faces from subject 24: five training images and one for testing (the last one). The training images must focus on what is expected to recognize, the frontal face in this case. The images do not present relevant lighting changes, however some present facial expression variations.

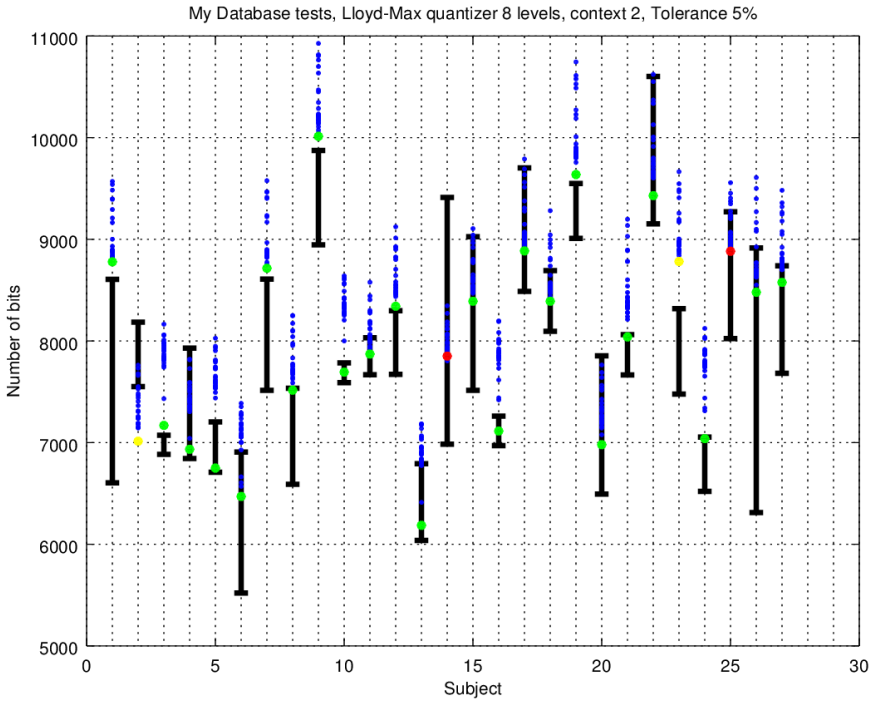


Figure 4.14: Recognition results using the created database for testing purpose. Images quantized with 8 levels, using a context of 2 pixels.

Positives	False Positives	Unrecognized
23	2	2

Table 4.9: Number of positives, false positives and unrecognized subjects, using the created database.

4.4 RECOGNITION USING PREDICTIVE METHODS

As referred on the previous chapter, there was a try to use predictive models on face recognition. The following tests have the same principle as the FCM ones: there is a model of a subject and it is expected that the minimum number of bits to encode the test image is achieved with the model belonging to the same subject.

Using the ORL database, for each model of each subject (9 first images of each 40 subjects), the

number of bits to encode all the test images was calculated, both for non-quantized and quantized (8 levels) images, as can be seen on Figure 4.15 and Figure 4.16 respectively. The number of positives/negatives can as well be observed on Table 4.10 and Table 4.11.

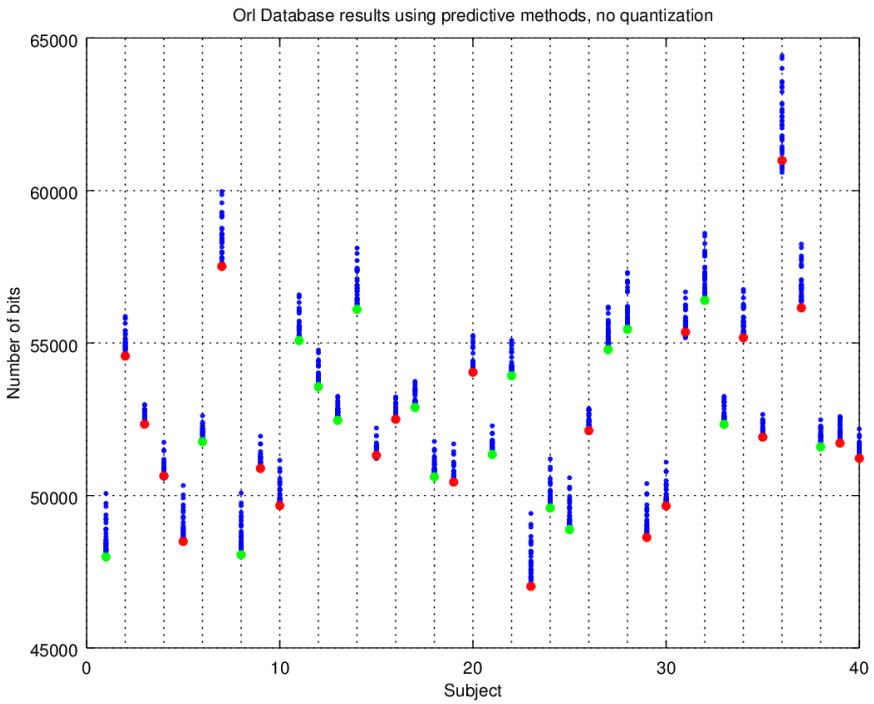


Figure 4.15: Recognition results on ORL Database using prediction.

Positives	**False Positives**
18	22

Table 4.10: Number of positives and negatives using predictive methods. Images are not quantized (8-bit gray-scale, 256 intensities).

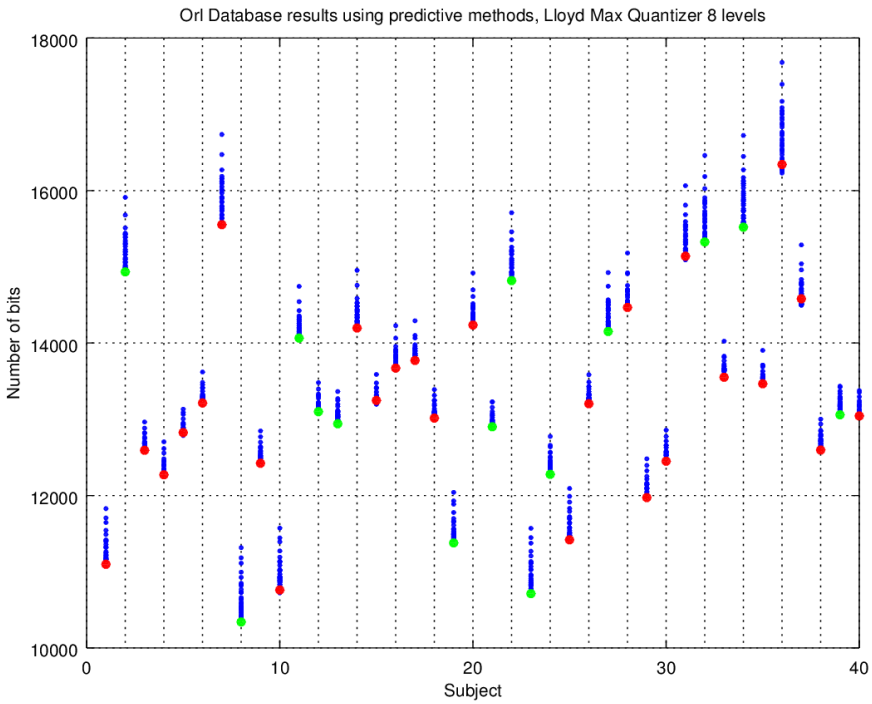


Figure 4.16: Recognition results on ORL Database (quantized 8 levels) using prediction.

Positives	**False Positives**
14	26

Table 4.11: Number of positives and negatives using predictive methods. Images are quantized with 8 levels.

However, contrarily to the FCM's case, these results prove to be inconclusive, because it is not clear that the minimum number of encoding bits of the image is obtained with the same model. Recall that on the case of the Finite Context Models, almost all of the test images obtained the minimum encoding bits with the model of the same subject. Therefore, it performed a lot better than the predictive methods, in terms of facial recognition.

4.5 FINAL REMARKS

From the tests executed and presented in this chapter, one can conclude that the present algorithm had a good performance on the recognition tests, in general. In the case of a real system, it is suitable to be used in a controlled environment, where the conditions are approximately the same, otherwise, because it is quite affected by changes to test images (as studied along this chapter), its behavior is not the ideal one.

Another important factor is the training set. The images to train the model must be the closest possible to the wanted test image. For example, if the application is required to recognize only frontal

faces, without some margin of rotation, the training set must be mainly constituted by frontal images; that is, the model must be trained with what one wants to recognize.

In this sense, namely on the FERET case, these images were wittingly chosen to a "worse case", and, despite the fact the results were a bit worse, there were cases where the algorithm did not behave that bad. Regarding the tolerance, it is suitable and wise to use on a real application (as long as a reasonable value is chosen), because it allows to differentiate those test images that, with high probability, belong to a subject, from those which do not.

When comparing this algorithm with the previously studied ones back on Chapter 2, it was shown that the Finite Context Models achieved (better case) 77% and 85% (ORL and the created database, respectively). Recall that the practical achieved results with the other algorithms was around 80 %. Besides, in terms of computational times, there is a big difference between the algorithms. Regarding the prediction time, this difference occurs because the prediction step is not the same: back on the Chapter 2 algorithms, this step consists on computing a test image with a single model of the entire database, while on the FCM the prediction is made by computing the same test image, but now with 40 different models.

CHAPTER 5

Conclusions and Future research

The research presented in this thesis had the main goal to develop and study a new possible and viable solution regarding the facial recognition problem, adding its own contribution to some research already performed on the subject. As one can see by the results on Chapter 4, it was shown that the proposed method, i.e., the compression algorithm based on the Finite Context Models, can actually be used for facial recognition, because the relation is evident, achieving, in some cases, very good recognition rates.

One of the facts to have into account is the training models, i.e., the training images. These images are quite important for achieving a good recognition, so they must be carefully chosen, meaning that the algorithm must be trained with what one wants to recognize; in other words, if one wants a frontal face recognition, the training set must be composed by that same frontal images. Note that the results have a very low number of false positives, i.e., test images which minimum number of bits was obtained with a different subject's model. On some cases, the unrecognized number is quite relevant. However, this can be improved exactly by the correct choice of the training set of images.

This algorithm proved to work under constrained environments. However its performance may not be as good on a non-constrained one, as shown by the lighting and rotation tests. This may be one of the next steps to take. Maybe some future research on the subject may become valuable to make the algorithm more flexible for other situations. To try to work around this problem, a possible solution would be to perform image normalization on the database's images. Some future research may prove valuable to increase the number of positives and decrease computational times; a possible approach can be the usage of several lower order (size of the context) Finite Context Models instead of a single one (recall that the time and memory required increase exponentially with the size of the context).

Another subject that may be a theme of research is the use of predictive methods. We have dedicated little time on this subject, because it was not the main objective, however, with the proper research, it can prove to be another way of perform facial recognition.

References

[1] R. Jafri and H. R. Arabnia, «A survey of face recognition techniques.», *Jips*, vol. 5, no. 2, pp. 41–68, 2009.

[2] D. L. Ruderman, «The statistics of natural images», *Network: computation in neural systems*, vol. 5, no. 4, pp. 517–548, 1994.

[3] K. Delac, M. Grgic, and S. Grgic, *Image compression in face recognition-a literature survey*. INTECH Open Access Publisher, 2008.

[4] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, «Face recognition: a literature survey», *ACM computing surveys (CSUR)*, vol. 35, no. 4, pp. 399–458, 2003.

[5] R. Brunelli and T. Poggio, «Face recognition: features versus templates», *IEEE transactions on pattern analysis and machine intelligence*, vol. 15, no. 10, pp. 1042–1052, 1993.

[6] R. Lienhart and J. Maydt, «An extended set of haar-like features for rapid object detection», in *Image Processing. 2002. Proceedings. 2002 International Conference on*, IEEE, vol. 1, 2002, pp. I–900.

[7] D. Bouchaffra, «Conformation-based hidden markov models: application to human face identification», *Neural Networks, IEEE Transactions on*, vol. 21, no. 4, pp. 595–608, 2010.

[8] X. Liu, «Video-based face model fitting using adaptive active appearance model», *Image and Vision Computing*, vol. 28, no. 7, pp. 1162–1172, 2010.

[9] H. K. Ekenel, J. Stallkamp, and R. Stiefelhagen, «A video-based door monitoring system using local appearance-based face models», *Computer Vision and Image Understanding*, vol. 114, no. 5, pp. 596–608, 2010.

[10] R. Chellappa, C. L. Wilson, and S. Sirohey, «Human and machine recognition of faces: a survey», *Proceedings of the IEEE*, vol. 83, no. 5, pp. 705–741, 1995.

[11] I. Kemelmacher-Shlizerman and R. Basri, «3D face reconstruction from a single image using a single reference face shape», *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 2, pp. 394–405, 2011.

[12] M. F. Hansen, G. A. Atkinson, L. N. Smith, and M. L. Smith, «3D face reconstructions from photometric stereo using near infrared and visible light», *Computer Vision and Image Understanding*, vol. 114, no. 8, pp. 942–951, 2010.

[13] Y. Wang, J. Liu, and X. Tang, «Robust 3D face recognition by local shape difference boosting», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, pp. 1858–1870, 2010.

[14] A. M. Bronstein, M. M. Bronstein, R. Kimmel, and A. Spira, «3D face recognition without facial surface reconstruction», *Technion-Computer Science Department Technical Report*, 2003.

[15] C. Beumier, «3D face recognition», in *Industrial Technology, 2006. ICIT 2006. IEEE International Conference on*, IEEE, 2006, pp. 369–374.

[16] M. Turk and A. Pentland, «Eigenfaces for recognition», *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[17] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, «Eigenfaces vs. fisherfaces: recognition using class specific linear projection», *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 711–720, 1997.

[18] P. Viola and M. Jones, «Rapid object detection using a boosted cascade of simple features», in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, IEEE, vol. 1, 2001, pp. I–511.

[19] P. Viola and M. J. Jones, «Robust real-time face detection», *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[20] T. Ahonen, A. Hadid, and M. Pietikäinen, «Face recognition with local binary patterns», in *Computer vision-eccv 2004*, Springer, 2004, pp. 469–481.

[21] A. Malkapurkar, R. Patil, and S. Murarka, «A new technique for lbp method to improve face recognition», *International Journal of Emerging Technology and Advanced Engineering*, 2011.

[22] P.-C. Hsieh and P.-C. Tung, «A novel hybrid approach based on sub-pattern technique and whitened pca for face recognition», *Pattern Recognition*, vol. 42, no. 5, pp. 978–984, 2009.

[23] L. Vu, A. Alsadoon, P. Prasad, A. Monem, and A. Elchouemi, «Face recognition template in photo indexing: a proposal of hybrid principal component analysis and triangular approach (pcaata)», in *2016 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, IEEE, 2016, pp. 177–180.

[24] M. Chihaoui, W. Bellil, A. Elkefi, and C. B. Amar, «Face recognition using hmm-lbp», in *Hybrid Intelligent Systems*, Springer, 2016, pp. 249–258.

[25] L. Torres and J. Vilá, «Automatic face recognition for video indexing applications», *Pattern recognition*, vol. 35, no. 3, pp. 615–625, 2002.

[26] Z. Li, D. Gong, X. Li, and D. Tao, «Aging face recognition: a hierarchical learning model based on local patterns selection», *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 2146–2154, 2016.

[27] A. Dahmouni, N. Aharrane, K. El Moutaouakil, and K. Satori, «Face recognition using local binary probabilistic pattern (lbpp) and 2d-dct frequency decomposition», in *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)*, IEEE, 2016, pp. 73–77.

[28] A. N. Kolmogorov, «Three approaches to the quantitative definition ofinformation'», *Problems of information transmission*, vol. 1, no. 1, pp. 1–7, 1965.

[29] T. I. Dix, D. R. Powell, L. Allison, J. Bernal, S. Jaeger, and L. Stern, «Comparative analysis of long dna sequences by per element information content using different contexts», *BMC bioinformatics*, vol. 8, no. 2, p. 1, 2007.

[30] A. Lempel and J. Ziv, «On the complexity of finite sequences», *IEEE Transactions on information theory*, vol. 22, no. 1, pp. 75–81, 1976.

[31] D. Salomon, *A concise introduction to data compression*. Springer Science & Business Media, 2007.

[32] K. Sayood, *Introduction to data compression*. Newnes, 2012.

[33] D. Salomon, *Data compression: the complete reference*. Springer Science & Business Media, 2004.

[34] D. Pratas and A. J. Pinho, «On the detection of unknown locally repeating patterns in images», in *Image Analysis and Recognition*, Springer, 2012, pp. 158–165.

[35] A. J. Pinho, D. Pratas, and S. P. Garcia, *Complexity profiles of DNA sequences using finite-context models*. Springer, 2011.

[36] A. J. Pinho and P. J. Ferreira, «Image similarity using the normalized compression distance based on finite context models», in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, IEEE, 2011, pp. 1993–1996.

[37] N. Karimi, S. Samavi, S. R. Soroushmehr, S. Shirani, and K. Najarian, «Toward practical guideline for design of image compression algorithms for biomedical applications», *Expert Systems with Applications*, vol. 56, pp. 360–367, 2016.

[38] M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi, «The similarity metric», *Information Theory, IEEE Transactions on*, vol. 50, no. 12, pp. 3250–3264, 2004.

[39] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications*. Springer Science & Business Media, 2013.

[40] M. J. Weinberger, G. Seroussi, and G. Sapiro, «Loco-i: a low complexity, context-based, lossless image compression algorithm», in *Data Compression Conference, 1996. DCC'96. Proceedings*, IEEE, 1996, pp. 140–149.

[41] X. Wu, «Lossless compression of continuous-tone images via context selection, quantization, and modeling», *IEEE Transactions on Image Processing*, vol. 6, no. 5, pp. 656–664, 1997.

[42] B. Carpentieri, M. J. Weinberger, and G. Seroussi, «Lossless compression of continuous-tone images», *Proceedings of the IEEE*, vol. 88, no. 11, pp. 1797–1809, 2000.

[43] `http://rnd.azoft.com/developing-face-recognition-system-convolutional-neural-network/`, 2015.

[44] `http://www.codeproject.com/Articles/441226/Haar-feature-Object-Detection-in-Csharp/`.

[45] `http://docs.opencv.org/3.0-last-rst/_images/at_database_small_sample_size.png`.

[46] `http://bytefish.de/blog/local_binary_patterns/`.

[47] `http://www.ual.es/~vruiz/Docencia/Apuntes/Coding/Image/02-LOCO/index.html`.

[48] `http://openhighschoolcourses.org/mod/book/tool/print/index.php?id=10838`.

[49] `http://speechpro-usa.com/files/image/news/VKL.JPG`.

[50] `http://vision.ics.uci.edu/images/fun/IMG_1183_augmented_reality_faces1.jpg`.

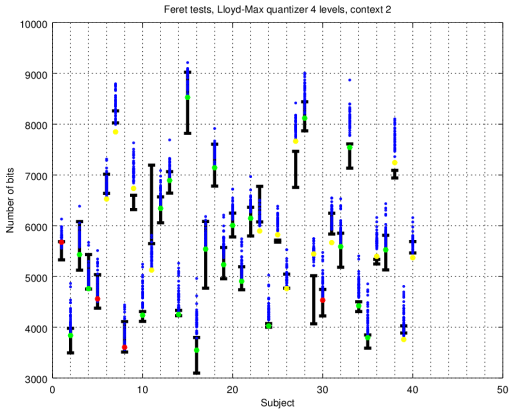
Appendix

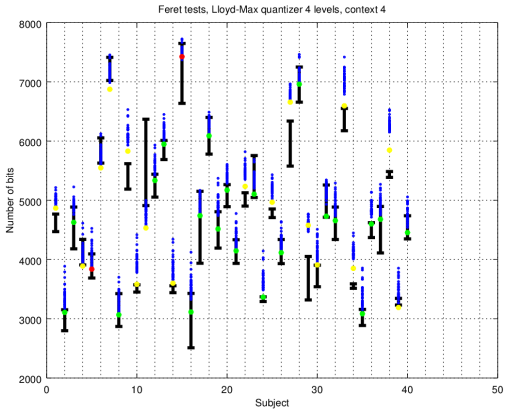
APPENDIX A - DETAILED RESULTS

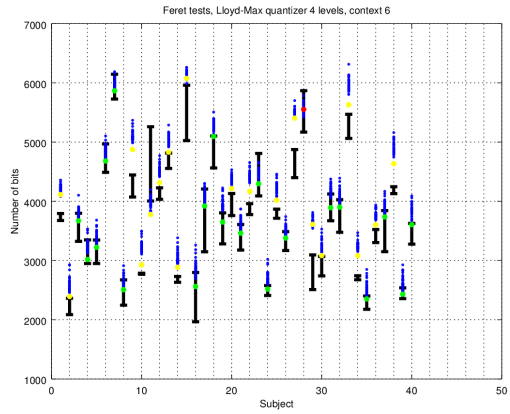
This section presents the detailed results of facial recognition achieved by both algorithms (Finite Context Models and predictive methods). The plots corresponding to the several situations described on Chapter 4 can be consulted, some of them were already presented on that same chapter, but not all of them could have been on the thesis main text.

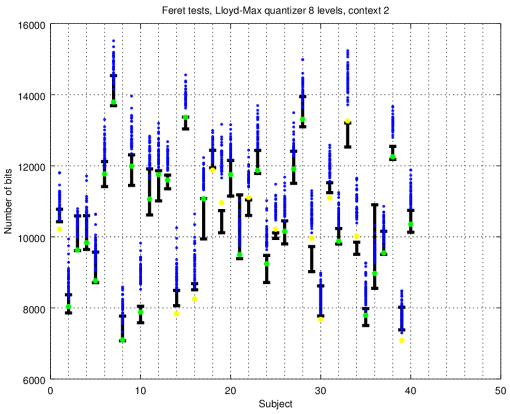
As can be seen, each plot title indicates clearly to which situation it corresponds.

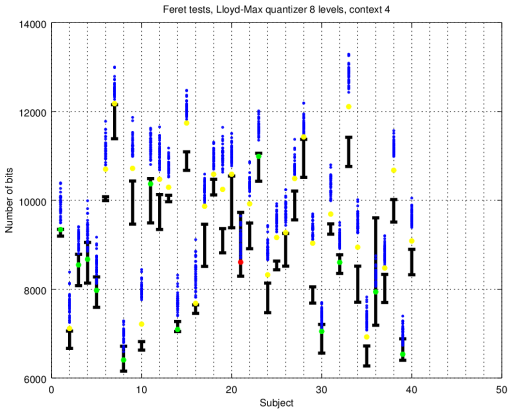
FERET TESTS

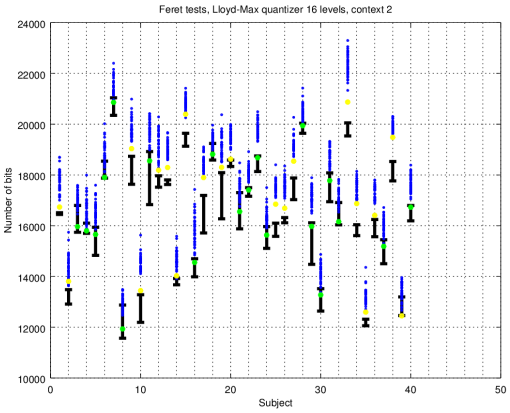


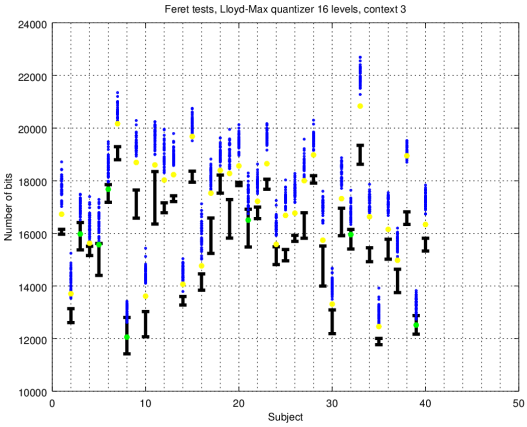



Feret tests, Lloyd-Max quantizer 4 levels, context 6
Number of bits
Subject

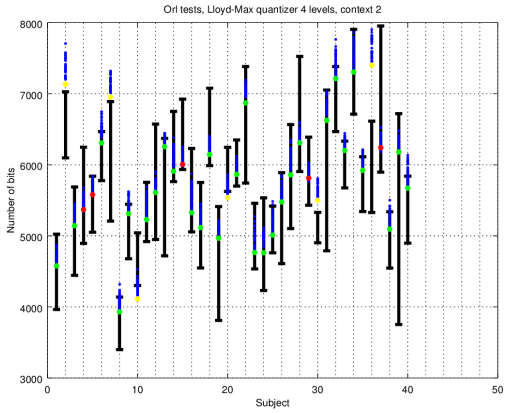

Feret tests, Lloyd-Max quantizer 8 levels, context 2
Number of bits
Subject

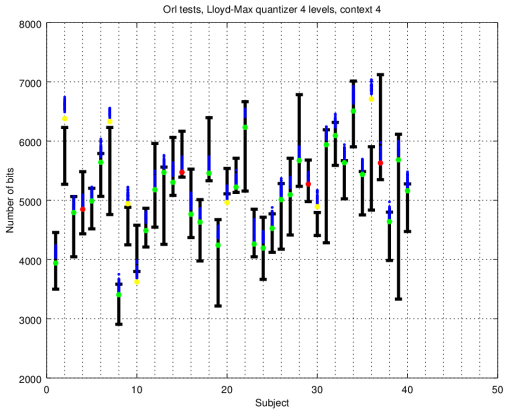

Feret tests, Lloyd-Max quantizer 8 levels, context 4
Number of bits
Subject

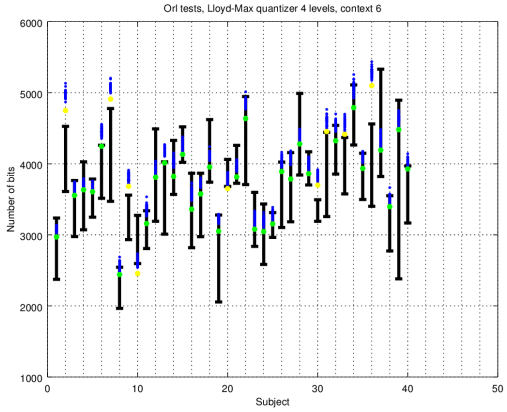

Feret tests, Lloyd-Max quantizer 16 levels, context 2
Number of bits
Subject

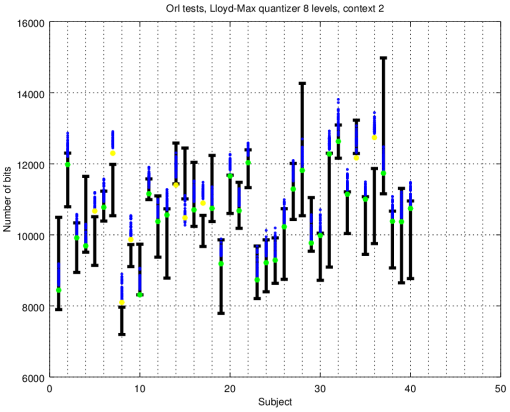

Feret tests, Lloyd-Max quantizer 16 levels, context 3
Number of bits
Subject

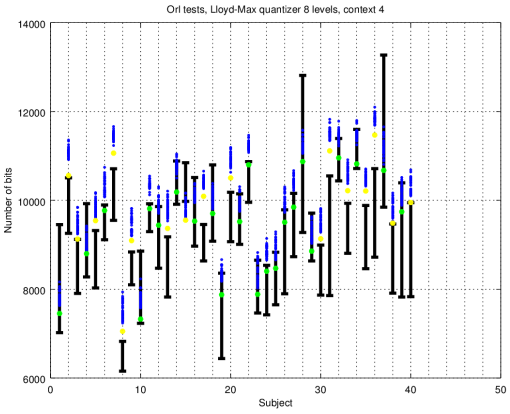
ORL TESTS

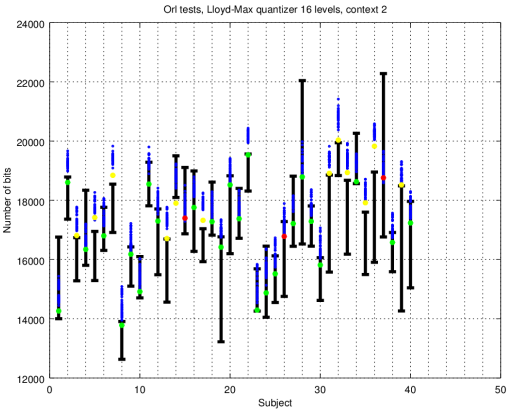

Orl tests, Lloyd-Max quantizer 4 levels, context 2
Number of bits
Subject

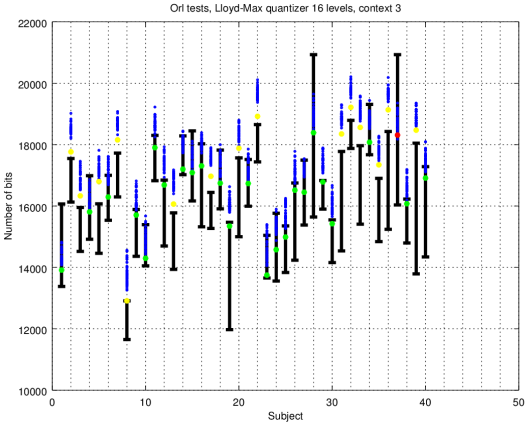

Orl tests, Lloyd-Max quantizer 4 levels, context 4
Number of bits
Subject


Orl tests, Lloyd-Max quantizer 4 levels, context 6
Number of bits
Subject

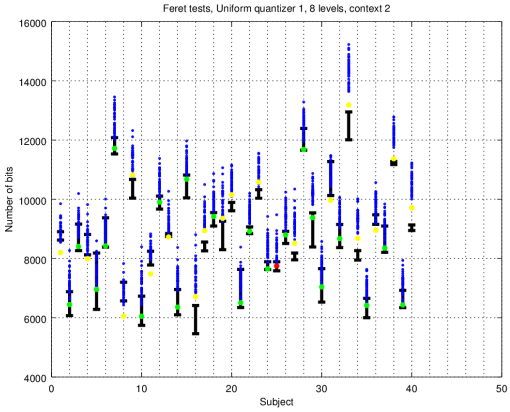

Orl tests, Lloyd-Max quantizer 8 levels, context 2
Number of bits
Subject

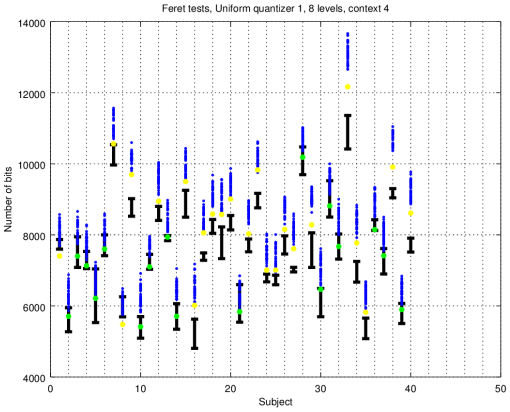

Orl tests, Lloyd-Max quantizer 8 levels, context 4
Number of bits
Subject

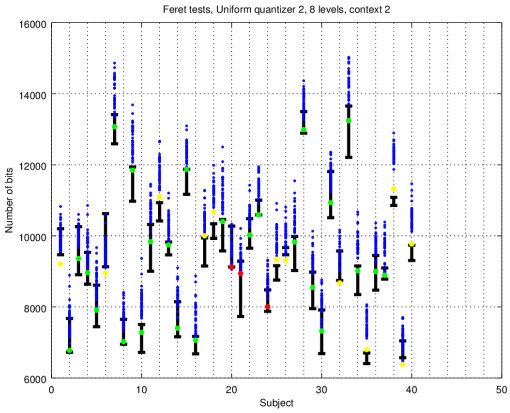

Orl tests, Lloyd-Max quantizer 16 levels, context 2
Number of bits
Subject

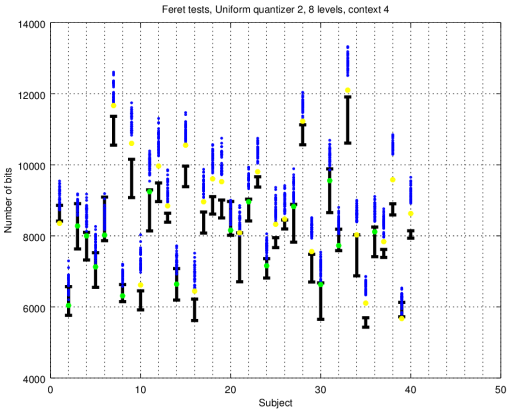


UNIFORM QUANTIZATIONS TESTS

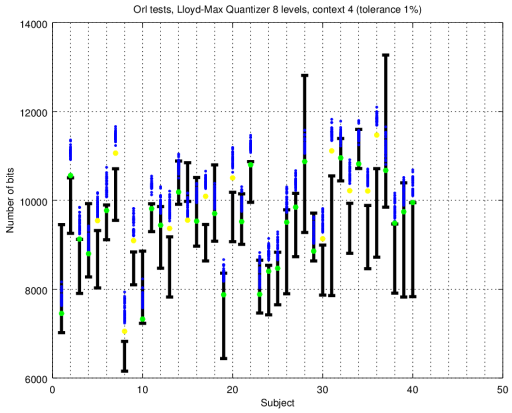


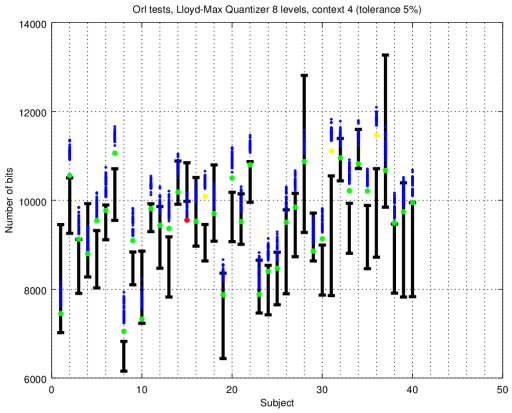


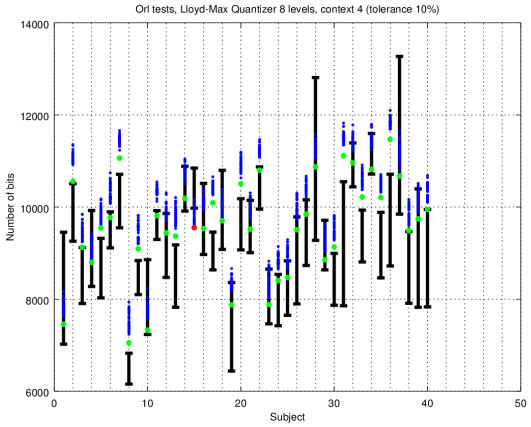




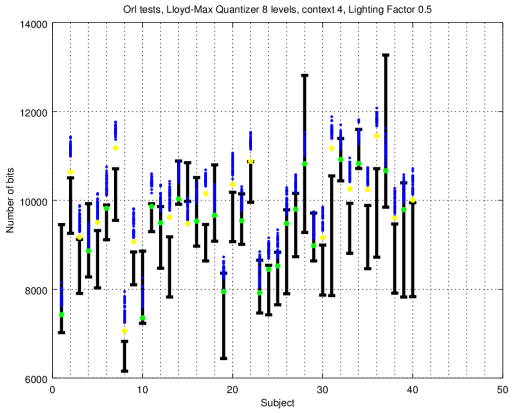
TOLERANCE TESTS

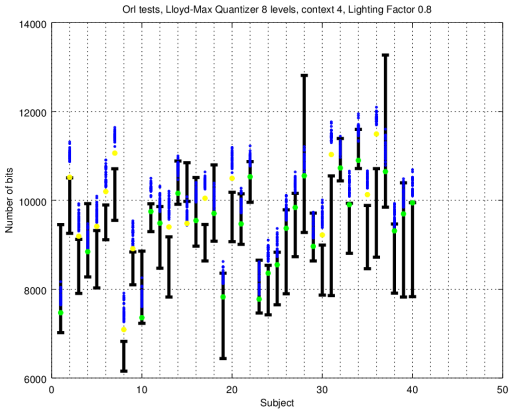


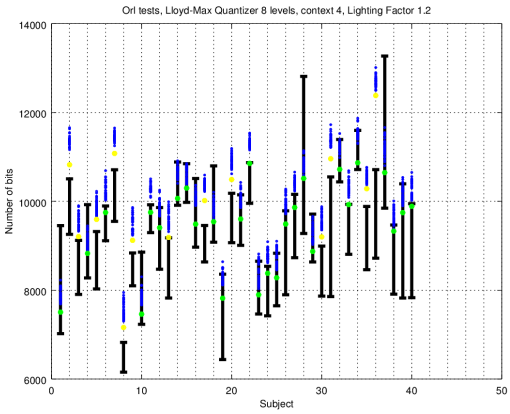


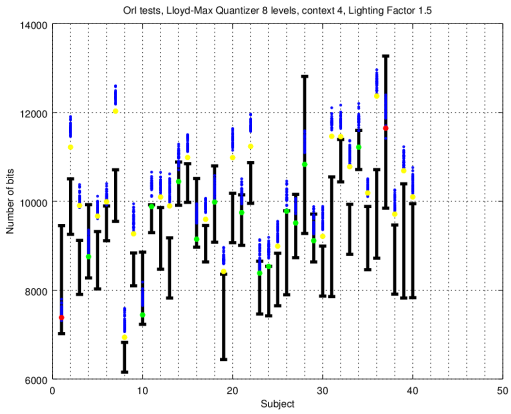


LIGHTING TESTS

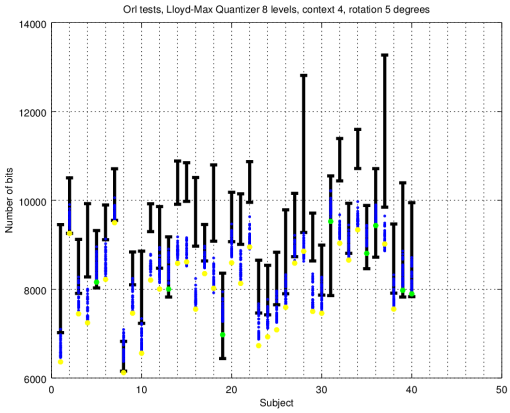


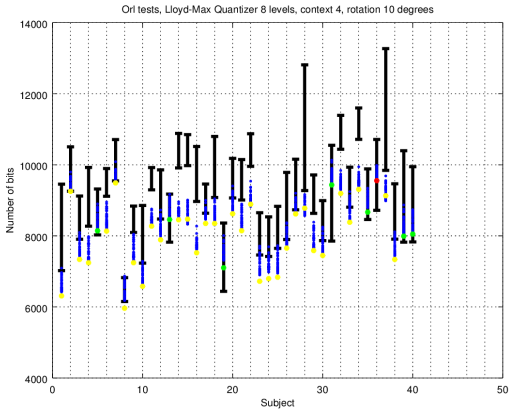


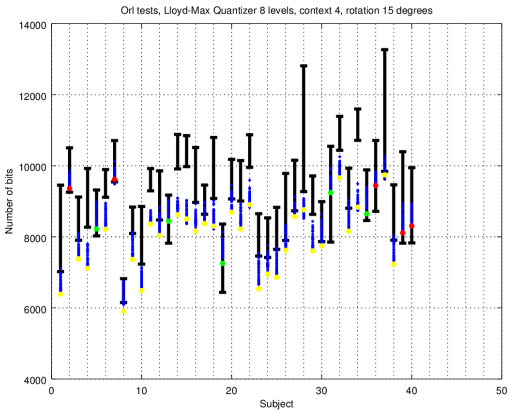


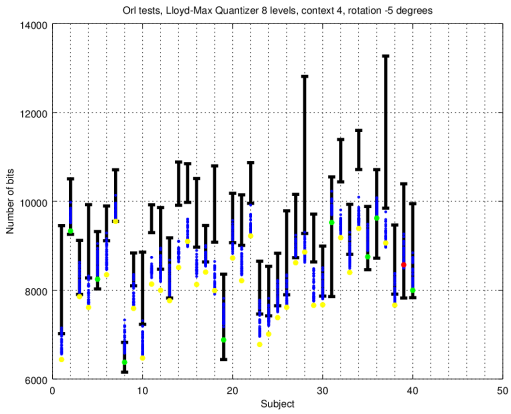


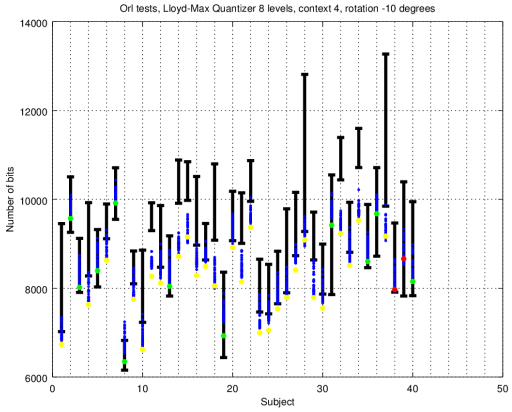
ROTATION TESTS

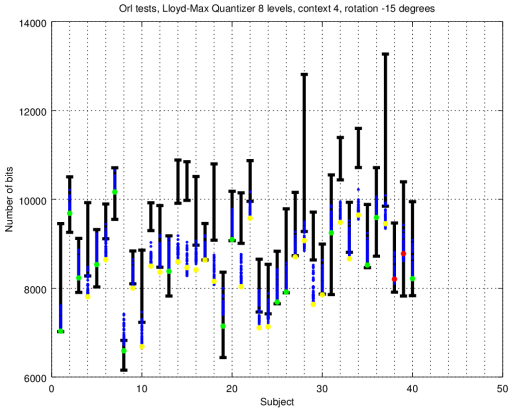




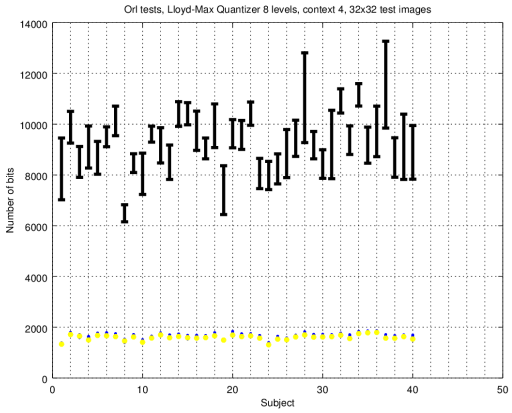


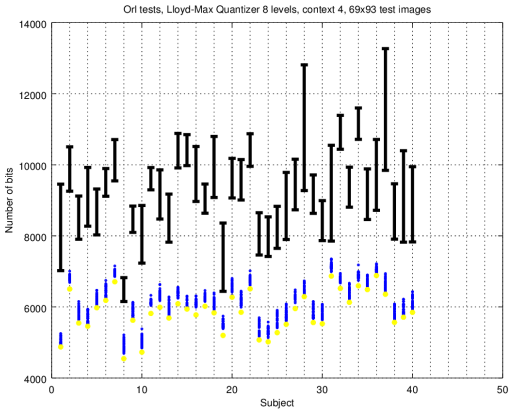


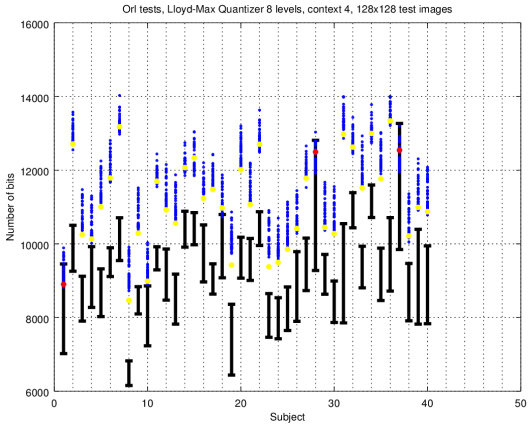

Orl tests, Lloyd-Max Quantizer 8 levels, context 4, rotation -10 degrees
Number of bits
Subject


Orl tests, Lloyd-Max Quantizer 8 levels, context 4, rotation -15 degrees
Number of bits
Subject

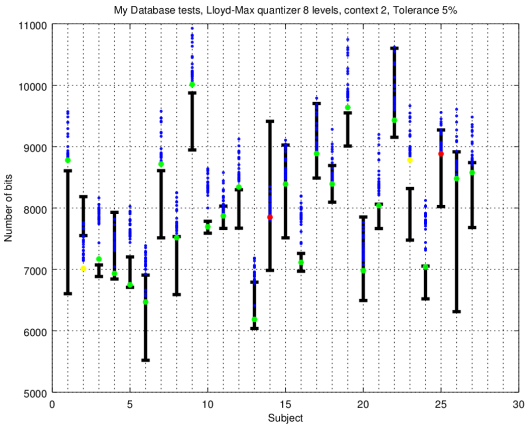
SCALE TESTS


Orl tests, Lloyd-Max Quantizer 8 levels, context 4, 32x32 test images
Number of bits
Subject

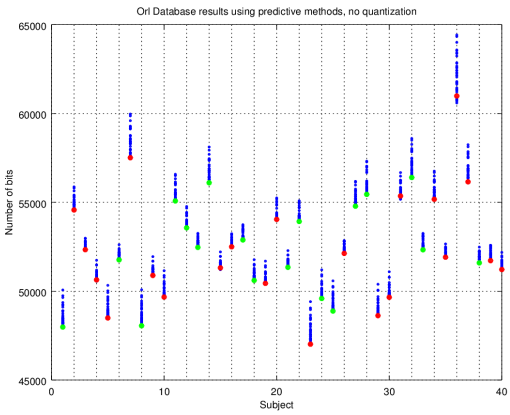

Orl tests, Lloyd-Max Quantizer 8 levels, context 4, 69x93 test images
Number of bits
Subject

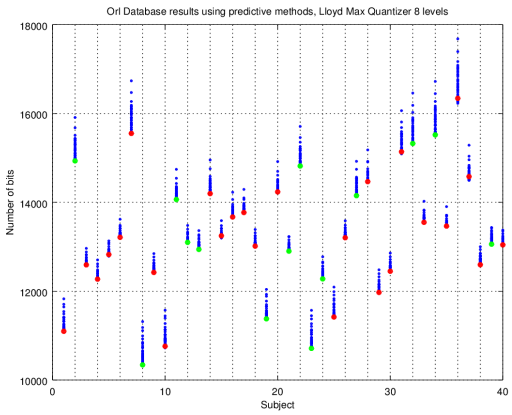

Orl tests, Lloyd-Max Quantizer 8 levels, context 4, 128x128 test images
Number of bits
Subject

TEST REGARDING THE CREATED DATABASE



PREDICTIVE METHODS TESTS





APPENDIX B - USER MANUAL

The present section resumes how the created libraries can be used to perform facial recognition.

GENERATING THE MODELS

First thing to do is to choose a database to perform facial recognition. It must be constituted by several subjects, each one with some training images (each subject has its own directory inside the database's directory). To train the model of a certain subject, there is the `train` executable, which allows to choose the context's size, either one or two dimensional:

```
Usage:   ./train [options]
options: -n context_size (previous horizontal pixels)
         -p context_size (two dimensional)
         -f folder_path
         -h (show help)
example: ./train -p 4 -f Databases/orl_quant/train/s1/
```

Listing 3: `Train` executable for models generation.

After executing the above example command, a **model.xml** file will be created at the specified folder *folder_path*, creating a 4-th (*context_size*) order Finite Context Model (two dimensional context); this file contains all the information about the model, as already explained back on Chapter 3. A bash script was developed to allow the training of the entire database at once.

PERFORMING RECOGNITION

After generating the models for the entire database, the `predict` executable allows to try to recognize a test image on a given database.

```
Usage:   ./predict [options]
options: -d database_path
         -i image_path
         -h (show help)
example: ./predict -d Databases/orl_faces/train/ -i test.pgm
```

Listing 4: `Predict` executable for recognition on a database.

The above example calculates the number of bits needed to encode the image *image_path* with all the database's models present on *database_path*. The output of this program is either a name of a subject from the database (which can be positive or false positive) or the Unrecognized label (corresponds to an image of a subject which is not on the database).

CHECKING THE NUMBER OF ENCODING BITS

Another available option besides the facial recognition itself, is to calculate the number of bits needed to encode an image, either using Finite Context Models or predictive methods.

USING FINITE CONTEXT MODELS

```
Usage:    ./fcm [options]
options: -n context_size (previous horizontal pixels)
         -p context_size (users chosen pixels)
         -i image_path
         -m model_path
         -r (reset table)
         -h (show help)
example: ./fcm -n 3 -m Databases/orl_faces/train/s1/ -i test.pgm
```

Listing 5: Finite context models to get the number of encoding bits.

This program outputs the number of bits needed to encode the image *image_path*, which can be achieved by two different ways: based on an already trained model *model_path* or based on one (or more) image(s) passed with the $-i$ option (in this case, the depth *context_size* of the model has to be passed).

USING PREDICTIVE METHODS

```
Usage:    ./predictor [options]
options: -n prediction_scheme
         -i image path
         -r (reset table)
         -h (show help)
example: ./predictor -n 8 -i test.pgm
```

Listing 6: Predictive methods to get the number of encoding bits.

This program outputs the number of bits needed to encode image *image_path* using predictive methods with a prediction scheme selected by the $-n$ option (8 is the LOCO-I). Can either be used with no prediction model (just one $-i$ option) or used with one/several images on the model. The prediction error is plotted.